快速上手

介绍

这一章作用是教你如何尽可能快的使用 MPLAB ICE 仿真器,这里假设你已经知道一些计算机硬件术语,如 Windows 操作系统,MPLAB IDE 软件,通用仿真器操作.

要点

这一章讨论:

- MPLAB ICE系统组成
- 安装MPLAB ICE
- 设置MPLAB ICE
- 使用MPLAB ICE

MPLAB ICE系统组成

MPLAB ICE 系统由这几部分组成(图 1):

1. 仿真器主机
2. 连接仿真器主机与 PC 的并行电缆
3. 电源供电电缆
4. 带电缆的处理模块
5. 将处理器模块连接到转换座的器件适配器
6. 将器件适配器连接到目标系统的转换座
7. 逻辑探针连接器

深圳市粤原点科技有限公司
(Microchip Authorized Design Partner)指定授权
总部地址：深圳市福田区福虹路世贸广场C座1103座
Add： Room 1103,Block C,World Trade Plaza,9Fuhong Road,Futian District Shen Zhen City
电话(tel) ：86-755-83666321,83666320,83666325
传真(fax) ：86-755-83666329
Web: www.origin-gd.com 或是 www.LZmcu.com
E-mail：03@LZmcu.com abc85185@163.com
联系人：王小姐,汤小姐
在线咨询：QQ:46885145 MSN:ivy660@hotmail.com
7x24小时在线产品咨询:13798484366

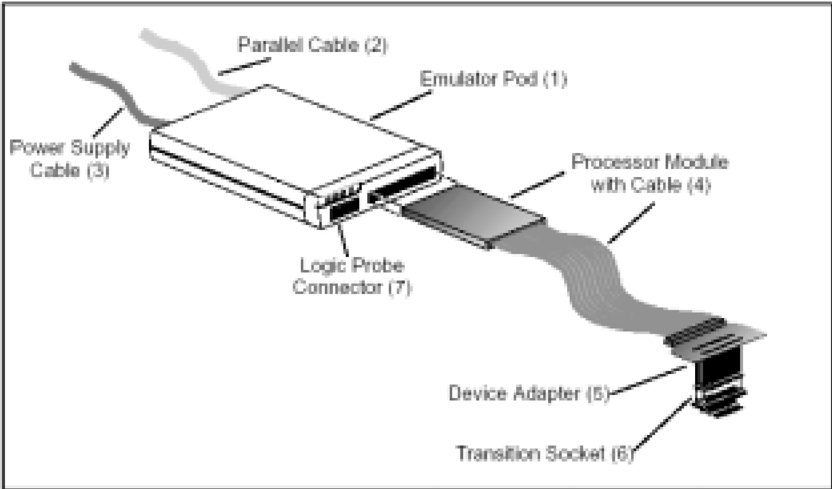


Figure 1: MPLAB ICE Emulator System

安装 MPLAB ICE

硬件安装

按照这节介绍的步骤安装仿真器硬件

WARNING	这时 PC 和 MPLAB ICE 的电源都应关闭

• **准备一未使用的 LPT 端口**

WARNING	使用MPLAB ICE 与一安全钥匙,外部磁盘驱动,或其它并行设备(如:打印机,扫描仪)相连会导致那个设备永久的损坏.

- **使用并行电缆将 MPLAB ICE 主机连接到并行口.**

  将并行电缆的一端连接到 MPLAB ICE 主机的并行连接器上,另一端连接到 PC 机的 LPT 端口.保证并行电缆的两端都已连好.

- **连接逻辑探头.**

  将逻辑探头插入仿真器主机前端的逻辑探头连接器里.

- **安装处理器模块**

  将处理器模块牢固的插入 MPLAB ICE 主机的前端.要取出处理器模块,应在处理器模块前端突出地方的后面各放一个手指,然后用力平稳的拉出.**切不可拉电缆**.

- **接上电源**

  在完成这步前,确认 MPLAB ICE 主机 on/off 开关是在"O"还是在"off"位置.将电源接线的两端分别插入电源插座和 MPLAB ICE 主机的背后.

- **如果通过 MPLAB ICE 系统使用目标板,则需安装器件适配座.**

  将器件适配座接到处理器模块电缆的末端.然后,将器件适配座插入目标板上的转换座.

- **打开系统组件.**

  为了防止任何子系统或目标应用部件的损坏,系统组件的上电应按照下列顺序:

  1. 打开 PC.
  2. 通过将仿真器主机开关拨到'I'来打开仿真器.
  3. 给目标应用电路供电.

  将系统组件关闭按照上述相反的顺序.

软件安装

要安装 MPLAB IDE 软件,请参考 MPLAB 用户手册的安装指导 (DS51025).

安装问题

如果你对硬件安装仍有困难,请参考关于安装章节的更多细节(第一章),或参考常见问题章节(第八章).如果你在安装软件时有问题,请参考 MPLAB 用户手册(DS51025).

设置 MPLAB ICE

开始使用 MPLAB

安装完 MPLAB IDE 软件后,执行文件 MPLAB.EXE.对于使用 MPLAB 的更多信息,请参考 MPLAB 用户手册(DS51025)和文件 README.LAB.

设置开发模式

打开 MPLAB IDE 的开发模式对话框来设置 MPLAB ICE 仿真器.选择 *Option>Development Mode*. 通过点击对话框的每一栏和下面说明的设置选项来设置开发模式.点击 **Apply** 接受设置,点击 **OK** 接受设置并关闭开发模式对话框.

- **Ports**选项: 在主机PC上显示关于LPT端口的信息.从下拉列表中选择一可用LPT端口要决定LPT端口是如何配置到

你的系统,可以点击**Query Port Info**.MPLAB ICE推荐运行模式为双向模式(PS/2,EPP,和ECP类型).

- **Tools**选项: 显示开发模式和器件信息.选择MPLAB ICE仿真器开发模式,并从下拉列表中选定仿真型号.

WARNING	如果有其它的设备连在并口上,请不要选择 MPLAB ICE 仿真模式,否则会对那个设备造成永久性的损害.

处理器的下面是关于 MPLAB ICE 仿真器仿真这款芯片时的任何限制的简要描述.要查看更详细的描述,点击 **Details**.

- **Power** 选项:显示系统电源信息.选择你想系统从那里获得电源;从它自身的电源(来自仿真器)或从目标系统(来自目标板).当选择目标板供电时,低电压仿真可被使用.
- **Clock** 选项:显示振荡器类型和频率信息.要使用板载时钟,选择振荡器类型并输入 32kHz 至 40MHz 之间的一个频率值.要使用目标板时钟,首先你必须在 **Power** 标签设置目标板供电,然后选择使用目标板时钟,这样它的实际频率会被计算出并显示出来.

除了上面的设置,你可能会根据你的应用或设备进行更多的改变.

- **Memory** 选项:使用这个标签设置要使用的存储器配置.这个标签并不是对所有处理器都能使用的.为了使片外存储器能被使用,处理器模式(*Options>Development Mode*, **Configuration** 标签)必须设置到 Microprocessor 或 Extended Microcontroller.
- **Configuration** 选项:使用这个标签设置看门狗定时器和/或处理器模式,如外部和内部存储器.处理器模式并不是对所有处理器都可用的.
- **Pins** 选项: 使用这个标签设置引脚,如使能/禁止主清除(MCLR).
- **Break Options** 选项: 使用这个标签来改变全局断点和跟踪点环境选项.

建立一个项目

通过选择 *Project>New Project* 来创建一个新项目.查找或为这个新项目创建一个目录,然后给这个项目命名(如 newprog.pjt).关于创建和使用项目的更多信息,请参考 MPLAB 用户手册(DS51025).

建立项目时可能的问题

如果你对开始使用 MPLAB 或建立一个新项目有困难,请参考 MPLAB 用户手册(DS51025).如果你对设置开发模式有困难,请参考关于设置的更详细章节(第四章),或参考常见问题章节(第八章).

使用 MPLAB ICE

MPLAB ICE 提供了各方面的工具来仿真和调试一个应用.MPLAB ICE 提供了一套基本的在线调试工具,包括运行,中止,复位和单步执行的能力.更多的工具可以提供更高级的调试能力.

MPLAB ICE 基本特性

MPLAB ICE 基本特性包括:

- 可下载功能
- 处理器存储器方式
- 状态栏和工具栏
- 软件断点
- 命名软件断点
- 触发输入输出

基本MPLAB ICE仿真器特性都内建在MPLAB IDE软件中.关于这些特性更多的信息,请参考MPLAB用户手册(DS51025).

MPLAB ICE高级特性

MPLAB ICE高级特性包括:

- 复杂触发
- 代码覆盖
- 跟踪窗口

这些特性会在下面的章节中分别讨论.关于这些特性更详细的信息,请参考第六章.

使用复杂触发

MPLAB ICE具有高度灵活和功能强大的触发功能.一个触发可以是下列事件的一个组合:

- 硬件断点,与/或
- 捕捉跟踪存储器.

在一个周期内能完成一个事件的捕捉并进行系统状态的描述.

另外,当触发发生时,可以产生一个外部信号.这对于其它模拟设备与MPLAB ICE的同步是很有用的.

复杂触发可以通过选择*Debug>Complex Trigger Settings*来指定.根据你的选择,复杂触发设置对话框会不同:

1. 存储器访问

- **程序存储器** – 如果是指定对程序存储器访问,触发可以发生在一条指令的取指时或一个表读/表写操作(对于具有表读/表写能力的PIC).
- **数据存储器** – 如果是指定数据存储器,触发会发生在一个读或写操作上.

> **注意:** 处理器模块不支持数据监测(12位内核处理器模块),也不支持数据存储器的触发.

2. 事件

- **顺序事件** – 如果每件事件必须按顺序发生来产生触发,那么选择**顺序事件.**注意对于顺序触发,仿真器会等待**事件1**满足,然后才会继续**事件2**,依次到最后一个**触发**事件.
- **所有事件** – 如果每个事件必须按任意顺序发生来产生触发,则选择**所有事件**.这个触发会在最后事件发生时产生.
- **任意事件** – 如果任意一个事件都会产生触发,那么选择**任意触发**.

- **事件间定时** – 选择**事件间定时**来指定一个开始和中止事件.这个事件选择使用在跟踪存储器窗口的连接处.在定时标志发生器开始递增之前,可以指定达到2个事件和开始事件顺序发生.
- **过滤跟踪** – 选择**过滤跟踪**来指定会被跟踪存储器窗口捕捉到的事件.在开始收集被指定事件前,有高至3个有资格的事件能被指定顺序发生.要执行连续过滤跟踪,查看**无限事件**检查窗口.

当执行过滤跟踪,忽略FNOP周期不应被指定,否则,跟踪缓存器会捕捉到引起触发的周期之后执行的那个周期.

> **注意:** 由于忽略FNOP周期不应和过滤跟踪一起指定,预取指会导致误触发.

同样,不推荐在数据存储器访问上执行过滤跟踪,因为在触发指定后被跟踪数据会是一个周期后被描述.

> **注意:** 当多个事件触发时,注意在数据存储器访问上的触发会与引起触发的指令偏移两个周期.

点击**Help**按钮,会打开在线帮助文件,并引导你进行复杂触发设置.

另外,一个单独的复杂触发断点可以通过点击鼠标右键来设置.在一个源文件窗口,一个列表窗口,或程序存储器窗口中,通过点击和拖拽选择期望的行.然后在鼠标右键菜单中选择*Complex Trigger Break Point*.这样就能设置和应用一个单事件顺序触发,并在触发设置上忽略FNOP周期和中止.复杂触发对话框不一定要打开.

使用代码覆盖

代码覆盖提供了突显被访问代码部分的特性(取指,写或读).这条代码在程序存储器窗口中被高亮显示.

- 要使能代码覆盖,选择*Debug>Code Coverage*,并选择运行时使能/复位或使能/手动复位.如果选择运行时使能/复位,代码复位会在每次仿真开始时复位.
- 如果选择使能/手动复位,代码覆盖仅仅在复位(第六章)被执行时才被复位
- 选择*Options>Environment Settings>Color*,为跟踪点文本选择一种颜色.这会是代码覆盖高亮的文本颜色.
- 要关闭代码覆盖,选择*Debug>Code Coverage*,并点击禁止.

> **注意:** 代码覆盖和复杂触发是相互独立的.

使用跟踪存储器窗口

MPLAB ICE跟踪存储器窗口中包含从MPLAB ICE跟踪缓冲器上传信息.默认情况下,所有指令周期都是通过跟踪缓存器来捕捉的.复杂触发对话框可以用来控制哪些指令周期被捕捉到了(查看使用复杂触发).

- 跟踪缓存器可以通过选择*Window>Trace Memory*来看到.

> **注意:** 由于处理器信号的定时,数据信息会偏移一个周期.目标数据值实际上会偏移两个周期,但是用于显示目的时,跟踪缓存器显示会为延时的周期补偿一个周期.

能被显示的指令周期数可以达到32767.跟踪存储器窗口对于每个执行周期包含下列信息:

- 周期数 – 相对于触发或中止点的周期位置.相应的触发周期会标为蓝色,并且被编为周期0.
- 地址(**Addr**) – 被从程序存储器取指的指令的地址.
- 操作数(**Op**) – 被取指的指令.
- 标识(**Label**) – 与程序存储器地址相关的标识.
- 指令(**Instruction**) – 未汇编的指令.
- 源数据地址(**SA**) – 如果适用的话,源数据的地址.

- 源数据值(**SD**) – 如果适用的话,源数据的值.
- 目标数据地址(**DA**) – 如果适用的话,目标数据的地址.
- 目标数据的值(**DD**) -如果适用的话,目标数据的值.
- 外部输入(**Ex**) – 外部输入的值.
- 定时标志(周期或秒) – 定时标志值.

注意: 当使用基于12位PIC单片机内核的处理器模块时,源和目标数据地址和值是不可用的.

- 选择*Data>Reload*来强迫上载数据,无论跟踪缓存器当前收集到什么值.这可以让你看到你的程序执行的地方.如果你的触发一直不出现的话,这常常会很有用.

使用问题

如果你在使用MPLAB ICE的特性时有任何问题,请参考关于特性章节(第五章或第六章)更多的细节,或者参考常见问题章节(第八章).

通用信息

介绍

第一章包括了一些通用的信息,在使用 MPLAB ICE 仿真器之前了解这些信息会很有用.

要点

你将从这章了解到的信息包括:

- 关于这个指南
- 授权登记
- 推荐阅读
- 常见问题
- Microchip网站
- 开发系统客户公告服务
- 客户支持

关于这个指南

文档规划

这个文档描述了如何使用MPLAB ICE作为一个开发工具来仿真和调试目标板上的固件. 手册规划如下:

- **快速上手** – 如何快速熟悉并使用MPLAB ICE.
- **第一章: 综述和安装** – MPLAB ICE是什么, 它怎样才能帮助你. 如何安装MPLAB ICE硬件和MPLAB IDE软件.
- **第二章: 指南 – PIC16CXXX** – 使用MPLAB ICE来仿真PIC16CXXX器件的指南.
- **第三章: 指南 – PIC18CXXX** - 使用MPLAB ICE来仿真PIC18CXXX器件的指南.
- **第四章: 通用设置** – 使用MPLAB IDE来设置MPLAB ICE.
- **第五章: 基本特性** – MPLAB ICE的基本特性的描述,(如,运行,中止,复位,单步等).
- **第六章: 高级特性** – MPLAB ICE的基本特性的描述,(如复杂触发,代码覆盖,跟踪).
- **第七章: 校验** – 如何确认MPLAB ICE的正确操作.
- **第八章: 常见问题** – 如何解决MPLAB ICE操作中的常见问题.
- **附录A: 调试技巧** – 如何使用MPLAB ICE来调试你的代码中常见问题.
- **附录B: 主机电气特性** – 仿真器主机的电气特性和描述.
- **附录C: 从PICMASTER移植** – MPLAB ICE与PICMASTER有什么不同.

这个指南中的使用惯例

这个手册使用下列文档惯例:

表一: 文档惯例

种类	描述	例子
代码(Courier字体):		
简单字符	样例代码 文件名和路径	#define START c:\autoexec.bat
角括号:<>	变量	<label>, <exp>
方括号[ ]	可选择见解	MPASMWIN [main . asm]
花括号和竖线字符: { | }	相互独立见解的选择 一个'或'选择	Errorlevel {0|1}
引用小写字母	数据类型	" filename "
椭圆…	用于暗示(但不显示)不相关例子的额外文本	List ["list_option… , "list_option"]
0xnnn	一个十六进制数,其中n是一个十六进制数字	0x FFFF, 0x007A
斜体字符	一个变量见解; 它可以是一个数据类型(小写字符),或一个特定的例子(大写字符).	Char isascii (char, *ch*) ;
接口(Arial 字体):		
有右箭头的带下划线的斜体文本	菜单栏的菜单选择	*File>Save*
粗体字符	用来点击的窗口或对话框按钮	**OK, Cancel**
带角括号<>的字符	键盘上的一个键	<Tab>, <Ctrl-C>
文档(Arial 字体):		
斜体字符	参考书	MPLAB IDE 用户指南

文档更新

所有的文档都会过期,这个用户指南也不例外.由于MPLAB IDE和其它Microchip工具都是不断发展来满足客户的需要,一些MPLAB IDE对话框和/或工具描述会与这个文档不同.请参考我们的网站www.microchip.com来获得最新的文档.

文档编号惯例

文档被编号为"DS". 编号写在每页的底端,在页码的前面. DS编号的编码惯例是: DSXXXXXA,

其中:

XXXXX = 文档编号.

A = 文档的版本等级.

授权登记

请填好附带的授权登记卡,并将它迅速邮寄出去. 这样,你会收到关于新产品更新的消息,

推荐阅读

这个用户指南描述了如何使用MPLAB ICE. 其它有用的文档列表如下.

README.ICE

关于使用MPLAB ICE的最新信息,请阅读MPLAB IDE CD-ROM里的README.ICE文件. README.ICE包含了一些更新信息,可能不包含在MPLAB ICE 用户指南中.

README.XXX

对于其它工具的最新信息,请参考产品的更多信息文件. 其它README文件查看MPLAB IDE目录.(如MPASM,文件名称为README.ASM)

MPLAB ICE处理器模块和器件适配头描述(DS51140)

对于MPLAB ICE 主机,不同的处理器模块和器件适配头请参考相关信息文档.

MPLAB ICE 转换座描述(DS51194)

关于转换座的信息请参考这个文档.

Microchip技术库CD-ROM(DS00161)

CD-ROM包含了所有Microchip产品广泛的应用笔记,数据手册和技术摘要. 要获得CD-ROM, 请联系当地Microchip销售服务商.

嵌入式控制手册Vol.1&2和嵌入式控制手册更新2000(DS00092,DS00167和DS00711)

这些手册包含了关于微控制器应用的丰富的信息.要获得这些文档,请联系当地的Microchip销售服务商.

常见问题

相关信息请参考第八章.

Microchip网站

Microchip提供网上在线支持.

Microchip使用网站是为了更方便客户获得文件和相关信息. 要访问这个网站,用户必须有一个网络浏览器并访问到英特网上.文件不提供FTP下载.

连接到Microchip互联网站

Microchip网站是: http://www.microchip.com

也可以通过FTP服务连接: ftp://ftp.microchip.com

通过网站,用户可以下载最新的资料,包括开发工具,数据手册,应用笔记,用户指南,文章和例程. 也包括各种商务信息,如Microchip销售办公室,分销商和工厂代表.其它数据资料如下所示:

- 最新Microchip发行资料
- FAQ技术支持
- 设计技巧
- 器件勘误

- Microchip参考程序成员列表
- 其它Microchip产品相关网站连接
- 产品,开发系统,技术信息参考
- 研讨会列表

开发系统客户公告服务

Microchip提供了客户公告服务来帮助我们的客户对Microchip产品保持最新的了解.一旦你订阅了列表服务项目,那么只要我们有更改,更新,废除或有相关的产品或开发工具的勘误,你就可以收到相关e-mail公告.

开发系统列表名称是:

- 编译器
- 仿真器
- 编程器
- MPLAB
- 其它工具

一旦你决定了你感兴趣的列表名称,你可以发信息到下列地址来订阅:

listserv@mail.microchip.com

按照下面的格式:

`Subscribe<listname>yourname`

下面是一个例子:

`Subscribe maplab John Doe`

要退订,可以发信息至:

listserv@mail.microchip.com

安装下面的格式:

`Unsubscribe<listname>yourname`

下面是一个例子:

`Unsubscribe mplab John Doe`

编译器

关于Microchip C编译器,连接器和汇编器.这些包括MPLAB C17, MPLAB C18, MPLINK和MPASM.

要订阅这个列表,发信息至:

listserv@mail.microchip.com

按照下面的格式:

Subscribe compilers yourname

仿真器

关于Microchip在线仿真器的最新信息.这些包括MPLAB ICE和PICMASTER.

要订阅这个列表,发信息至:

listserv@mail.microchip.com

按照下面的格式:

Subscribe emulators yourname

编程器

关于Microchip PIC单片机编程器的最新信息.这些包括PRO MATE II和PICSTART Plus.

要订阅这个列表,发信息至:

istserv@mail.microchip.com

按照下面的格式:

Subscribe programmers yourname

MPLAB

关于Microchip IDE的最新信息.这个列表主要集中在MPLAB IDE, MPLAB SIM,MPLAB IDE项目管理器和通用编辑调试特性.对于MPLAB IDE的编译器,连接器和汇编器,请订阅编译器列表.对于MPLAB IDE仿真器的信息,请订阅仿真器列表.对于MPLAB IDE编程器的信息,请订阅编程器列表.

要订阅这个列表,发信息至:

listserv@mail.microchip.com

按照下面的格式:

Subscribe mplab yourname

其它工具

由Microchip提供的其它开发工具的最新信息. 关于MPLAB IDE和它的集成工具的相关信息请参考其它邮件列表.

要订阅这个列表,发信息至:

listserv@mail.microchip.com

按照下面的格式:

Subscribe otools yourname

客户支持

Microchip产品的用户可以通过几个渠道获得帮助:

- 分销商或代理商
- 当地销售办公室
- 现场应用工程师(FAE)
- CAE
- 热线

客户可以打电话给分销商,代理商或FAE获得支持.当地销售办公室也为客户提供帮助.

第一章 综述和安装

1.1 介绍

这章将让你对 MPLAB ICE 系统有个总的了解,然后解释如何安装系统硬件和软件.

1.2 要点

这章讨论:

- 什么是MPLAB ICE
- MPLAB ICE 系统组成
- MPLAB ICE 如何帮助你
- MPLAB ICE 套件组成
- 安装 MPLAB ICE 硬件
- 给系统部件供电
- 安装 MPLAB IDE 软件

1.3 什么是 MPLAB ICE

MPLAB ICE 是一种用来仿真所有 PIC 单片机的在线仿真器.最新的仿真处理器提供全速仿真.并且在执行期间,指令和数据路径都是可视的.

MPLAB ICE 2000 执行基本功能,如运行,中止,单步,软件断点和指令地址跟踪,更多特性如数据监测,复杂触发,扩展跟踪和代码覆盖.

MPLAB ICE 支持集成到了 MPLAB 这个 Microchip 集成开发环境.MPLAB 为你的应用的开发和调试提供了一个桌面环境.

这个文档覆盖了 MPLAB ICE 仿真器的基本设置和操作.但它并没有包括 MPLAB IDE 的所有功能.

1.4 MPLAB ICE 系统组成

MPLAB ICE 系统由这些部分组成(如图 1.1):

1. 仿真器主机
2. 连接仿真器和 PC 的并行电缆
3. 电源电缆

4. 带电缆的处理器模块
5. 连接处理器模块到转换座的器件适配头
6. 连接器件适配头到目标系统的转换座
7. 逻辑探针连接器

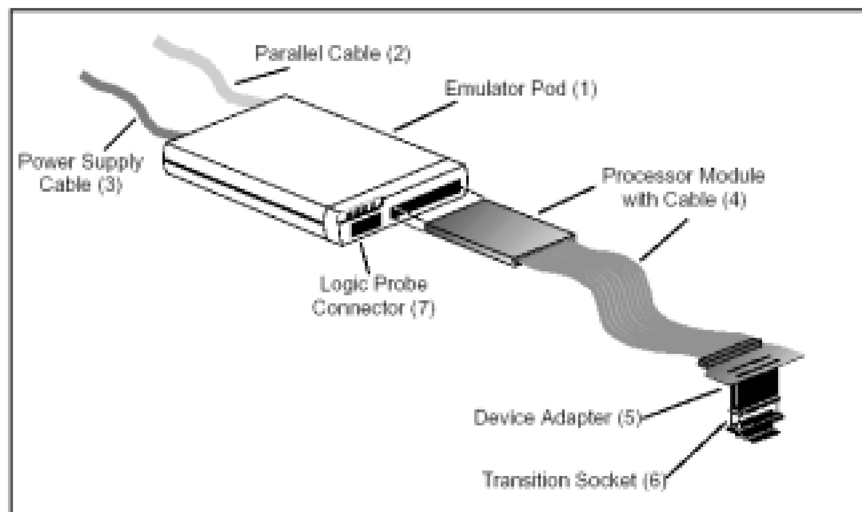


Figure 1.1: MPLAB ICE Emulator System

仿真器主机使用提供的电缆通过并行口连接到 PC.它包含了一般仿真器所具有的功能,如跟踪,中断和仿真.

处理器模块插入到仿真器主机前端的插槽中.它包含了所仿真器件或系列器件的硬件功能.

器件适配器放置在处理器模块电缆末端上.它连接转换座.转换座直接连接到目标系统.

逻辑探针连接器也是可使用的.逻辑探针可以连接在这儿,或单独的跳线可以连接到单独的连接器引脚.关于连接器和逻辑探针的更多信息请参考附录 B.

1.5 MPLAB ICE 如何帮助你

MPLAB ICE 允许你:

- 在你自己的硬件上实时调试你的应用.
- 硬件和软件断点调试.
- 测量事件之间的时间.
- 基于内部和/或外部信号设置断点.
- 内部文件寄存器监测.
- 全速达到 40MHz 仿真
- 在软件中选择振荡源
- 编程应用时钟速度
- 跟踪数据总线获得和定时标志事件

• 基于程序和数据总线事件和外部输入设置复杂触发.

注意: 有些处理器模块不支持数据总线事件监测

1.6 MPLAB ICE 套件组成

MPLAB ICE 仿真套件的组成,加上额外的硬件如图 1.2.

1. 用户指南
2. CD-ROM,包含 MPLAB IDE 软件和在线文档
3. 连接仿真器主机和 PC 的并行电缆
4. 仿真器主机
5. 电源和电缆
6. 仿真器支架---选配件
7. 带电缆的仿真器模块---选配件
8. 逻辑探头---选配件
9. 连接处理器模块到目标系统的器件适配器---选配件
10. 连接器件适配器到目标系统板的插针---选配件
11. 转换座---选配件



Figure 1.2: MPLAB ICE Emulator Kit and Optional Hardware

1.7 安装 MPLAB ICE 硬件

按照下列步骤安装仿真器硬件.

WARNING	这时 PC 和 MPLAB ICE 的电源都应关闭

- **准备一未使用的LPT端口**

WARNING	使用MPLAB ICE 与一安全钥匙,外部磁盘驱动,或其它并行设备(如:打印机,扫描仪)相连会导致那个设备永久的损坏.

建议 MPLAB ICE 直接通过并口运行,中间不要接开关盒或其它设备,因为 MPLAB ICE 是使用它自身的通讯协议.
如果这个并口已经在使用中,应单独为 MPLAB ICE 安装一个并口.

- **使用并行电缆将 MPLAB ICE 主机连接到并行口.**

  将并行电缆的一端连接到 MPLAB ICE 主机的并行连接器上,另一端连接到 PC 机的 LPT 端口.保证并行电缆的两端都连接好.

- **连接逻辑探头.**

  将逻辑探头插入仿真器主机前端的逻辑探头连接器里.

- **安装处理器模块**

  将处理器模块牢固的插入 MPLAB ICE 主机的前端.要取出处理器模块,应在处理器模块前端突出地方的后面各放一个手指,然后用力平稳的拉出.**切不可拉电缆**.

- **接上电源**

  在完成这步前,确认 MPLAB ICE 主机 on/off 开关是在”O”还是在”off”位置.将电源接线的两端分别插入电源插座和 MPLAB ICE 主机的背后.

- **如果通过 MPLAB ICE 系统使用目标板,则需安装器件适配座.**

  将器件适配座接到处理器模块电缆的末端.然后,将器件适配座插入目标板上的转换座.

1.8 给系统组件上电

MPLAB ICE 必须通过外部电源供电来运行.

为了防止任何子系统或目标应用部件的损坏,系统组件的上电应按照下列顺序:

1.8.1 打开系统组件

按照下面描述的顺序来给系统组件上电,以此来防止任何对子系统或用户目标应用部分的损害. 如果要使用低电压仿真,那么在使用这一特性前请先看第 1.8.2 节.

WARNING	如果不按照这些步骤,可能会对仿真系统或目标应用造成损害.

1. 组装 MPLAB ICE 仿真系统.

关于如何组装仿真系统,请参照快速上手章节或安装章节(第一章).

2. 开 PC.

3. 器主机开关拨到'I'来打开仿真器.

WARNING	在打开仿真器主机前,先插入处理器模块.千万不要在给主机上电后插入处理器模块.

4. 给目标应用电路供电.

MPLAB ICE 允许仿真处理芯片通过仿真器主机或目标系统来供电.这可以在 MPLAB 中按如下进行设置:

- 仿真器主机: *Options>Development Mode*, 电源选项, 选择处理器电源来自仿真器
- 目标系统: Options>Development Mode, 电源选项, 选择处理器电源来自目标板.

> **注意:** 当电源由仿真器主机来提供时,MPLAB ICE 提供给系统 10mA 电流.当电源由目标板提供时,MPLAB ICE 提供给系统 80 mA 电流.

当连接到目标系统,你应该注意目标应用的电压级别,即使你还没有给目标应用电路加电.这是由于保护二极管通过器件适配器的 Vcc 产生漏电流. 漏电流通常会小于 20mA.然而,如果目标应用正在使用一个电压调节器,那么应该注意,有些调节器在 V_{IN} 和 V_{OUT} 之间需要外部关断二极管的使用来进行翻转偏移保护.参考厂商的数据手册来了解更多信息.

1.8.2 打开系统组件 – 低电压仿真

MPLAB ICE 也支持低电压仿真,从 2.0V 至 4.5V. 在低电压仿真下,**电源必须由目标系统来提供.**

> **注意:** 当电源由仿真器主机来提供时,MPLAB ICE 提供给系统 10mA 电流.当电源由目标板提供时,MPLAB ICE 提供给系统 80 mA 电流.

按照下列步骤来给系统上电.

WARNING	如果不按照这些步骤,可能会对仿真系统或目标应用造成损害.

1. 组装 MPLAB ICE 仿真系统,但**还不要**将器件适配器插入目标板上的转换座中.

关于如何组装仿真系统,请参考快速上手章节或安装章节.

2. 打开 PC,并启动 MPLAB-IDE.
3. 通过将仿真器主机开关拨到'I'来打开仿真器.

WARNING	在打开仿真器主机前,先插入处理器模块.**千万不要**在给主机上电后插入处理器模块.

MPLAB 最初会从仿真器主机来给处理器模块供电,以保证正确的初始化.一旦系统初始化,你可以按照下一步所说的选择外部电源.

4. 设置 MPLAB.

从 MPLAB 选择 <u>*Option>Development Mode*</u>. 点击**端口**标签来设置 MPLAB ICE LPT 端口. 点击**工具**标签来为 MPLAB ICE 仿真器和你的目标处理器设置开发模式. 关于设置的更多信息,请看快速上手章节或设置章节.

最后,点击**电源**标签. 改变处理器电源来自仿真器为处理器电源来自目标板. 然后点击 **OK**.

由于目标板上没有电源,所以你会得到一条错误信息"处理器不能中止", 点击 **OK**.

WARNING	在将 MPLAB ICE 器件适配器连到目标应用系统前,你必须选择外部电源(来自目标板),否则会对目标应用系统造成损害.

5. 将器件适配器插到目标板的转换座中.
6. 给目标应用电路上电.
7. 在出现"处理器不能中止"错误对话框时,点击 **OK**.
8. 选择 *Options>Development Mode*. 你应该在开发模式对话框的**电源**标签栏看到低电压使能.

要使用目标时钟,点击**时钟**标签,选择使用目标板时钟.点击 **OK**.

1.8.3 关闭系统组件

1. 从目标应用电路移走电源.
2. 关闭仿真器电源.
3. 关闭 PC.

1.9 安装 MPLAB IDE 软件

安装 MPLAB IDE 软件,请参考 MPLAB 用户手册.

MPLAB IDE 软件以及 MPLAB ICE 仿真器在下列操作系统下能正常使用:

- Microsoft Windows 3.1x
- Windows 95/98
- Windows NT 3.51 或更高
- Windows 2000

第二章 指南-PIC16CXXX

2.1 介绍

在安装完 MPLAB ICE 硬件和 MPLAB 软件后,你就可以通过这个指南来开始工作了.

2.2 要点

这个指南包括:

- 硬件预览
- 运行MPLAB-IDE
- 设置开发模式
- 创建一个项目
- 建立项目
- 使用软件断点
- 使用命名软件断点
- 使用硬件断点
- 使用复杂触发
- 使用代码覆盖
- 总结

2.3 硬件回顾

这个指南中所使用的硬件设置见下面列表:

- **PC LPT Port:** LPT1,双向模式
- **MPLAB ICE 主机:** MPLAB ICE 2000
- **MPLAB ICE 处理器模块:** PCM16XE1

尽管你可能是使用LPT端口,而不是LPT1,但是,还是推荐你尽可能使用LPT1.而且,最好是使用双向模式.因为尽管兼容模式也能工作,但它会比较慢.

在这则指南中介绍了许多中档 PIC 单片机的处理器模块.其中,PIC16C74B 作为一个例子. 然而其它 PIC16CXXX 处理器只需作极小的修改就能使用.

2.4 运行 MPLAB

在安装完 MPLAB IDE 软件后,通过执行 MPLAB.EXE 来运行软件.

关于使用 MPLAB 的更多信息,请参考 MPLAB 用户指南和文件 README.LAB.

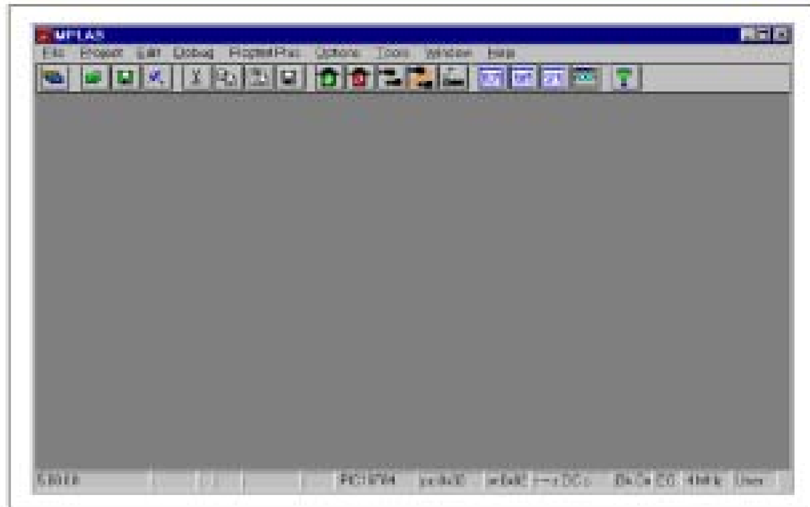


Figure 2.1: MPLAB IDE

2.5 设置开发模式

在 MPLAB IDE 软件中,打开开发模式对话框(*Options>Development Mode*)来设置 MPLAB ICE 仿真器.通过点击对话框的每个标签和下面的设置选项来设置开发模式.

2.5.1 端口标签

端口标签显示主机 PC 上空闲 LPT 端口的信息.

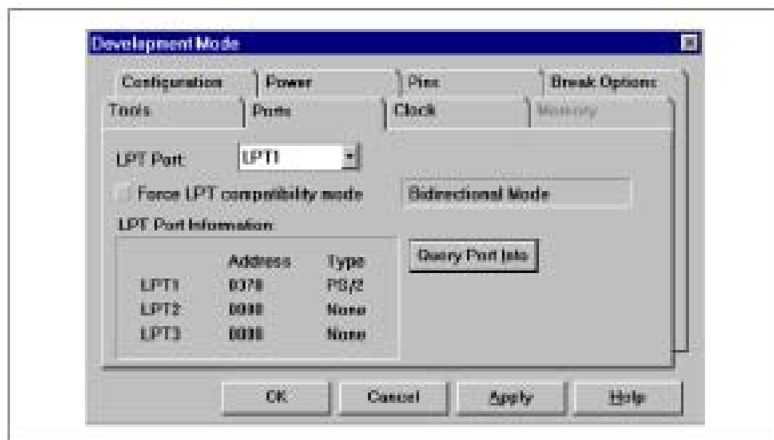


Figure 2.2: Development Mode Dialog – Ports Tab

从下拉列表中选择可供使用的 LPT 端口.要决定这个 LPT 端口是如何配置到你的系统,点击 **Query Port Info**.推荐 MPLAB ICE 的运行模式是双向模式(PS/2, EPP 和 ECP 类型).

点击 **Apply** 接受这个设置.

如果你还有任何关于 LPT 端口配置的问题,请参考设置章节或常见问题的细节.

2.5.2 工具标签

工具标签显示了开发模式和器件信息.

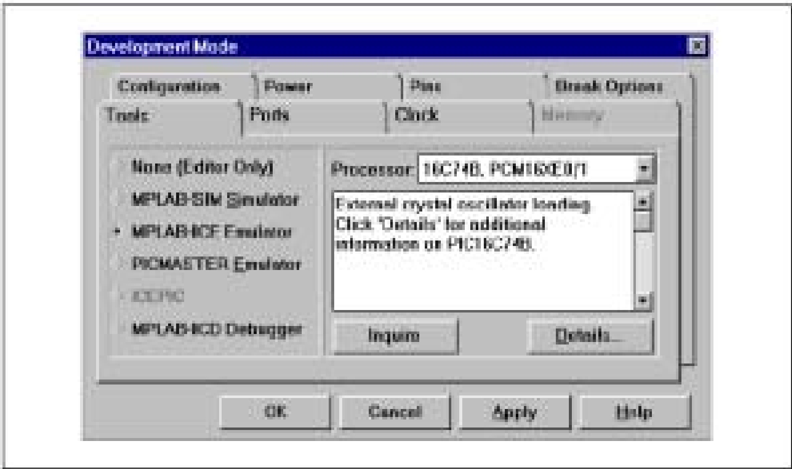


Figure 2.3: Development Mode Dialog – Tools Tab

在开发模式中选择 MPLAB ICE 仿真器

WARNING	如果有其它的设备连在并行口上,千万不要选择 MPLAB ICE 仿真模式,否则会对这个设备造成永久性的损害.

从下拉列表中选择 PIC16C74B 来仿真.在处理器的下面,是关于 MPLAB ICE 仿真器仿真这款型号时的一些限制的简单描述.更多细节可以点击细节进行查看.

点击应用接受这个设置.

2.5.3 中断选项标签

中断选项标签用来改变全局中断和跟踪点环境选项.默认值为下载清除断点,全局中断使能和中止冻结外设.在这则指南中,不检查中止冻结外设.

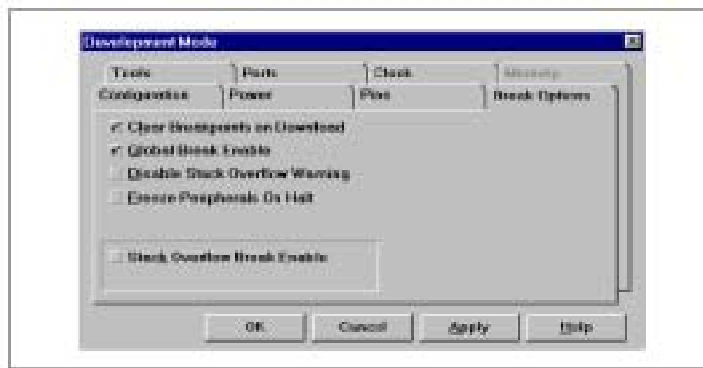


Figure 2.4: Development Mode Dialog – Break Options Tab

点击 **Apply** 接受设置**.**

2.5.4 其它标签

其它标签不需要再编辑.然而,你也许希望查看它们,使你熟悉那些选项.

- **电源**标签: 显示系统电源信息. 默认是处理器从仿真器供电.
- **时钟**标签: 显示振荡器类型和频率信息. 默认是使用板载时钟,振荡器类型为 LP,频率为 4.0MHz.
- **存储器**标签: 使用这个标签来设置被使用的存储器配置.对于选定的处理器,这个标签不能使用.
- **配置**标签: 使用这个标签来设置看门狗定时器和/或处理器模式.默认是看门狗定时器关闭.对于选定的处理器,处理器模式不可用.
- **引脚**标签: 使用这个标签来设置引脚,如使能/关闭主清除(MCLR).对于选定的处理器,这个选项不可用.

当你设置完毕,点击 **OK** 接受设置,并关闭开发模式对话框.

2.6 创建一个项目

使用 MPLAB IDE 软件开发一个项目最好的方法是创建一个项目.在这个指南中,你将要创建一个项目,名称为 `icetut16.pjt`.然而在这之前,你应该先建一个名字为 `icetut16` 的文件夹.你建的项目将放在这个文件夹中.同时,你应该将 MPLAB 安装目录下的 `icetut16.asm` 文件拷贝到这个文件夹中.

通过选择 *Project>New Project* 来创建一个新项目.新项目对话框会出现.

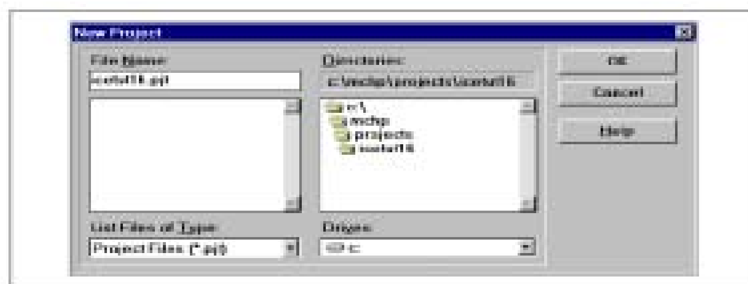


Figure 2.5: New Project Dialog

找到你创建项目的目录,然后给这个项目命名为 icetut16.pjt.点击 **OK** 关闭对话框并打开编辑项目对话框.



Figure 2.6: Edit Project Dialog – Hex File Only

在项目文件对话框中列出的一个文件是 icetut16[.hex], 这个文件将作为输出文件.

> **注意:** 如果你使用的 MPLAB IDE 不是最近才装的,(即,默认配置可能已经更改),你将需要:
>
> - 点击Hex文件来激活节点属性按钮
> - 点击这个按钮打开接到属性
> - 在这个对话框中选择MPASM作为语言工具
> - 点击**OK**关闭对话框并返回编辑项目对话框

紧挨着项目文件框是一组按钮,其中一个**添加节点**按钮处于激活状态.点击它打开添加节点对话框.

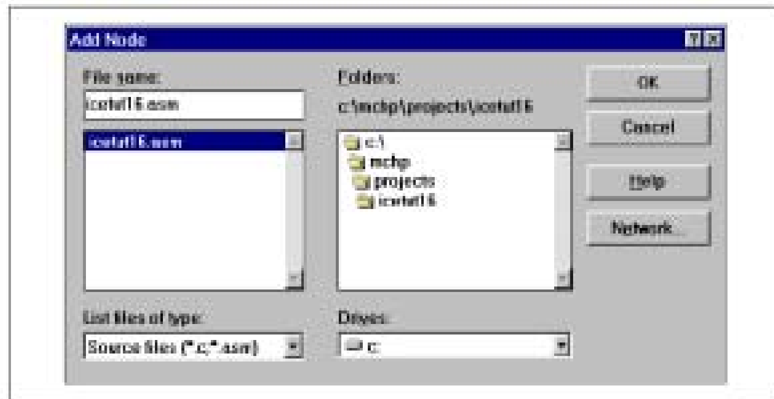


Figure 2.7: Add Node Dialog

找到你拷贝到项目目录中的 ASM 文件,然后点击它的名字选择它.如果你找不到这个文件,或者如果它丢失了,在对话框的文件名这一栏中输入 icetut16.asm.

点击 **OK** 关闭对话框并返回到编辑项目对话框.

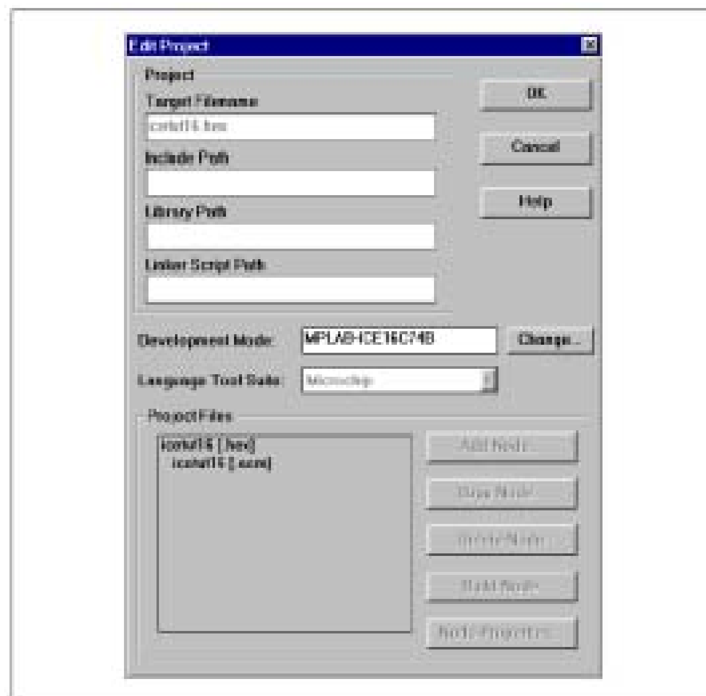


Figure 2.8: Edit Project Dialog – Hex and ASM Files

现在 ASM 文件 icetut16[.asm]应该列出在 Hex 文件 icetut16[.hex]下面.点击 **OK** 关闭对话框.

如果你能找到并拷贝文件 icetut16.asm 到项目目录中,你现在应该在 MPALB IDE 主窗口中,而没有其它窗口打开.这样可以进行下一步.

如果你不能找到这个文件,你现在应该看到 MPLAB IDE 主窗口中有一个打开的公文件窗口,名字为 Untitled.

Figure 2.9: MPLAB IDE with Untitled File Window

你可以通过 *File>Save As* 给这个文件窗口命个名.当 Save File As 对话框打开时,选择这个项目目录 icetut16,并输入 icetut16.asm 作为文件名.点击 **OK**. 现在你就给这个文件命了名,并准备开始在这个空窗口中输入源代码.

icetut16.asm 源代码:

```
        list        p=16c74b
        title       "PIC16C74B Tutorial - Flash PORTB LEDs"

        #include     <p16c74b.inc>

TEMP1   equu        0x20            ;Delay loop
TEMP2   equ         0x21            ;temp. variables
TEMP3   equ         0x22

        org         0x00            ;Reset Vector
        goto        START

        org         0x05            ;Start Program
START
```

```
        clrf    PORTB           ;Clear PortB (LED's off)

        bsf     STATUS, RP0     ;Select Bank 1
        clrf    TRISB           ;Set PortB as output
        bcf     STATUS, RP0     ;Select Bank 0

LOOP
        movlw   0xFF
        movwf   PORTB           ;Set PortB
        call    DELAY           ;Wait

        clrf    PORTB           ;Clear PortB
        call    DELAY           ;Wait

        goto    LOOP            ;Repeat

;*******************************************
;*   This routine is a software delay.     *
;*   Fosc = 1/Tosc; Tcycle = 4 x Tosc      *
;*   Delay = TEMP1xTEMP2xTEMP3xTcycle *
;*******************************************
DELAY
        movlw   0xFF
        movwf   TEMP1           ;TEMP1 = 255
        movwf   TEMP2           ;TEMP2 = 255
        movlw   0x07
        movwf   TEMP3           ;TEMP3 = 7
DLOOP
        decfsz  TEMP1, F
        goto    DLOOP

        decfsz  TEMP2, F
        goto    DLOOP

        decfsz  TEMP3, F
        goto    DLOOP

        return

        END
```

通过 File>Save 保存这个文件.然后通过 File>Close 或点击窗口右上角关闭文件.

2.7 建立这个项目

在这个指南中,建立项目也就是编译源代码.因为在项目中只有一个ASM文件.选择Project>Build All来建立这个项目.当完成时,建立结果窗口会出现.

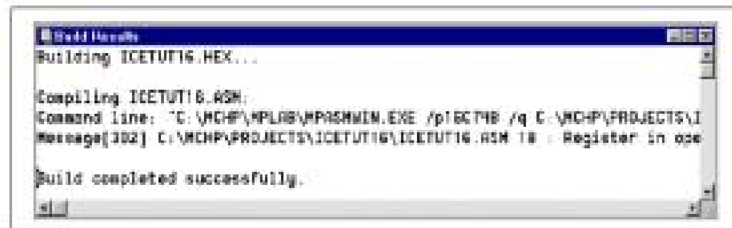


Figure 2.10: Build Results Window

如果你是自己输入的源代码,且建立失败,请检查你的拼写然后再尝试建立.

更多信息请参考 MPLAB 用户指南.

2.8 使用软件断点

现在你的项目已被建立好,并且代码成功编译成一个可执行(.hex)程序.你可以在 MPLAB ICE 上运行这个程序.

接着打开源代码文件.首先选择 File>Open 打开 Open Existing File 对话框.在目录列表中找到这个项目目录,选择 `icetut16.asm`. 点击 **OK**.

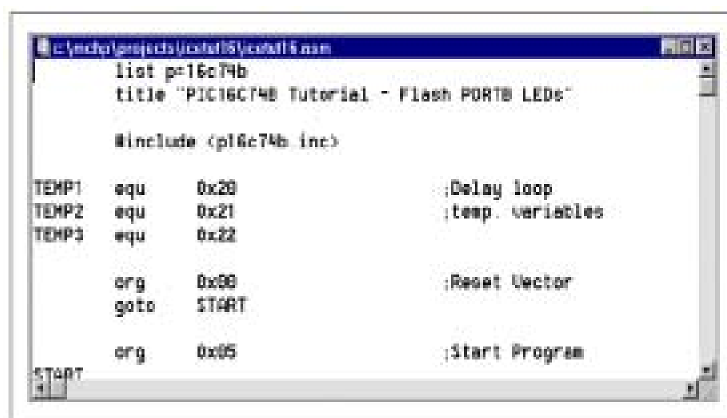


Figure 2.11: Source Code File Window

接着,通过选择 Window>Watch Windows>New Watch Window 打开一个新的观察窗口. 在添加观察窗口对话框滚动列表中找到 PORTB, 点击选中 PORTB 选项.点击 **Add**. PORTB 应该出现在观察窗口中.点击 **Close** 关闭添加观察窗口对话框.

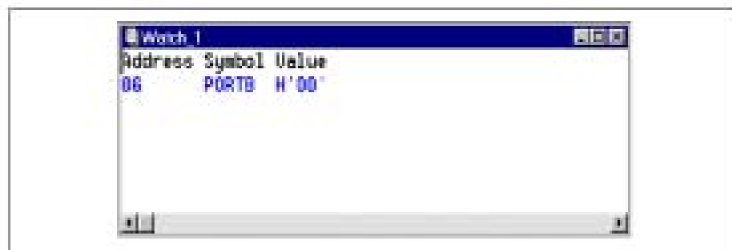


Figure 2.12: Watch Window

现在你可以在源代码文件窗口中设置一个软件断点.回到程序的主循环,在 movwf PORTB 这一行点击鼠标右键.会出现一个菜单.选择软件断点.这一行会改变颜色.如果你不喜欢这种颜色,你可以通过选择 *Options>Environment Setup* 来改变.点击颜色标签,选择一种不同的颜色用于断点文本.

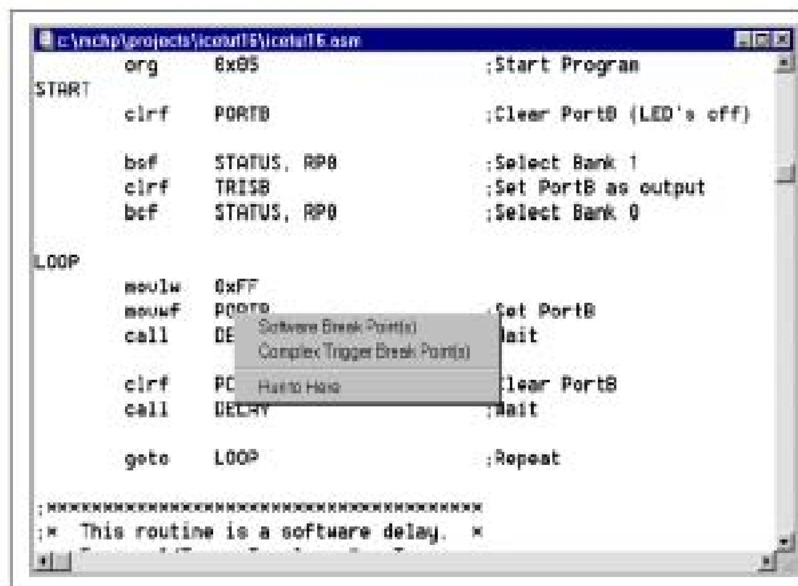


Figure 2.13: Set Software Break Point

通过 *Debug>Run>Run* 或点击工具栏上的绿色灯图标来运行程序. MPLAB IDE 窗口的底端状态栏会改变颜色,显示程序在运行.当程序达到断点时会中止.

通过选择 *Debug>Run>Step*,或按**<F7>**,或点击工具栏上的单步图标来执行单步.你应该看到观察窗口中 PORTB 的值已经从 H' 00 ' 变为 H' FF ',并且应该改变了颜色.

这演示了软件断点是如何在 MPLAB ICE 中工作的.在这行代码被执行之前,断点会中止仿真运行.然后单步执行一次,代码会被执行并显示在观察窗口中.

你也可以从鼠标右键菜单中选择 Run to Here,而不是 Software Break Point,来设置一个临时断点.

2.9 使用命名软件断点

MPLAB允许最多16个命名软件断点.这些断点可以被选择性的使能和关闭.以这种方式设置的断点可以随着项目一起保存.鼠标右键菜单断点不可以.

通过选择 Debug>System Reset 或点击工具栏上复位图标来复位程序.这会清除所有断点.

通过选择 Debug>Break Settings 打开断点设置对话框. 在对话框的开始栏上输入 0x000A,这样将 **break1** 断点指定在指令 movwf PORTB,然后点击 **Add**.

> **注意:** 要决定一条指令的地址,可以使用程序存储器窗口(Window>Program Memory).

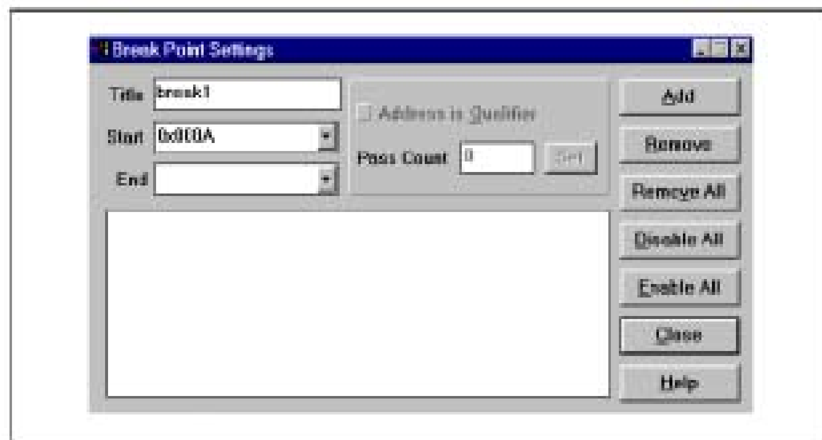


Figure 2.14: Break Point Settings Dialog

你应该在断点设置对话框对应 0x000A 的 **break1** 处于使能状态,在源程序窗口相应指令行应改变颜色. 点击断点设置对话框上的 **Close** 关闭窗口.

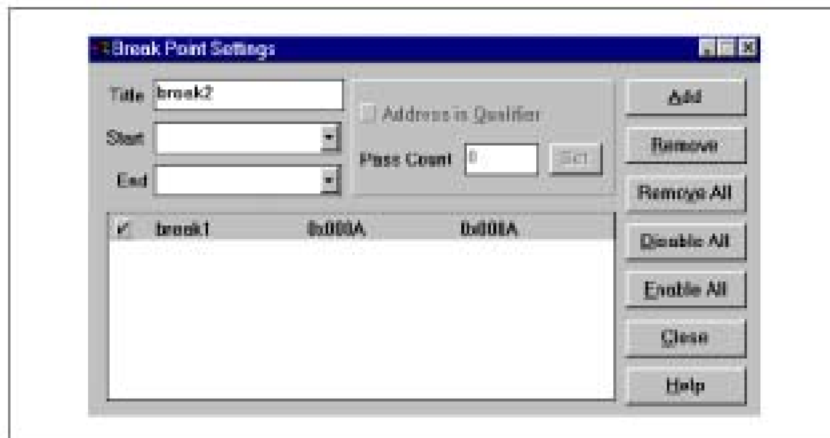


Figure 2.15: Break Point break1 Set

通过选择 *Debug>Run>Run* 或点击工具栏绿色灯图标来运行程序.MPLAB IDE 窗口底端的状态栏会改变颜色,表示程序正在运行.当程序运行到断点会中止.

单步运行,你可以看到观察窗口中的 PORTB 值从 H'00'变为 H'FF'.

2.10 使用硬件断点

除了在程序存储器地址设置软件断点外,也可以设置硬件断点.

通过选择 *Debug>System Reset* 或点击工具栏上复位图标来复位程序.这会清除所有断点.

跳到程序的主循环,在 movwf PORTB 上点击鼠标右键. 会出现一个菜单.这时选择复杂触发断点.这一行会改变颜色,如果你不喜欢这种颜色,你可以选择 *Options>Environment Setup*,点击 **Color** 标签,然后选择一种你喜欢的颜色.

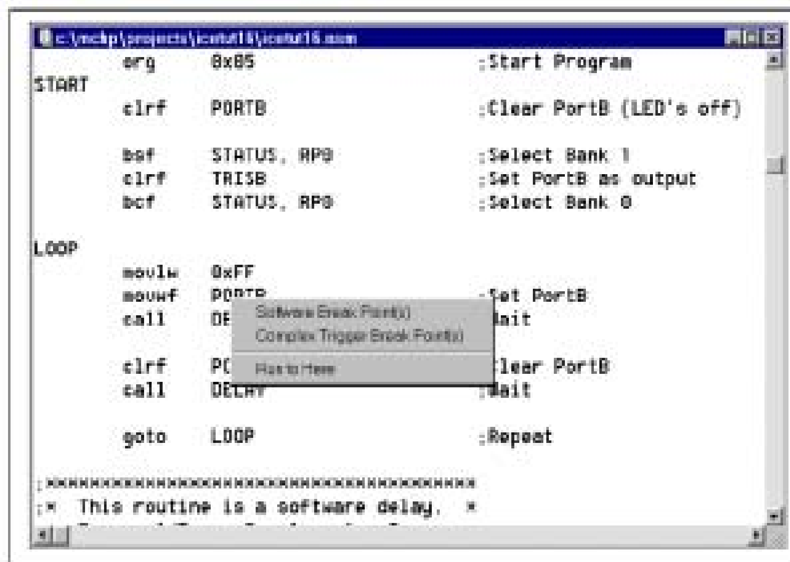


Figure 2.16: Set Hardware Break Point

通过选择 *Debug>Run>Run* 或点击工具栏绿色灯图标来运行程序.MPLAB IDE 窗口底端的状态栏会改变颜色,表示程序正在运行.当程序中止.它会回到原来的颜色.

注意,程序并不在你设断点的地方停止.你应该看到在 DELAY 程序的 movlw 0xFF 指令被加亮,这表示程序是在这条指令中止运行的.

这表示,当你使用硬件断点来中止运行时,在处理器被中止前,你可能会跳过一条或多条额外的指令.

2.11 使用复杂触发

设置复杂触发设置对话框可以完成:

- 一个硬件断点
- 一个跟踪存储器捕捉

当使用了这个对话框,这个项目的复杂触发设置就被保留了. 鼠标右键菜单复杂触发断点不可用.

在这个指南中,复杂触发会首先用作一个硬件断点,然后作为一个跟踪存储器捕捉.

2.11.1 复杂触发作为一个硬件断点

通过选择 *Debug>System Reset* 或点击工具栏上复位图标来复位程序.这会清除所有断点.

要设置复杂触发,可以通过选择 *Debug>Complex Trigger Settings* 打开复杂触发设置对话框.设置复杂触发 **Ignore FNOP Cycles** 和 **Halt on Triigger,**这样它就可以象鼠标右键复杂触发断点一样工作了(如,在地址 0x000A 作为一个单事件顺序触发).

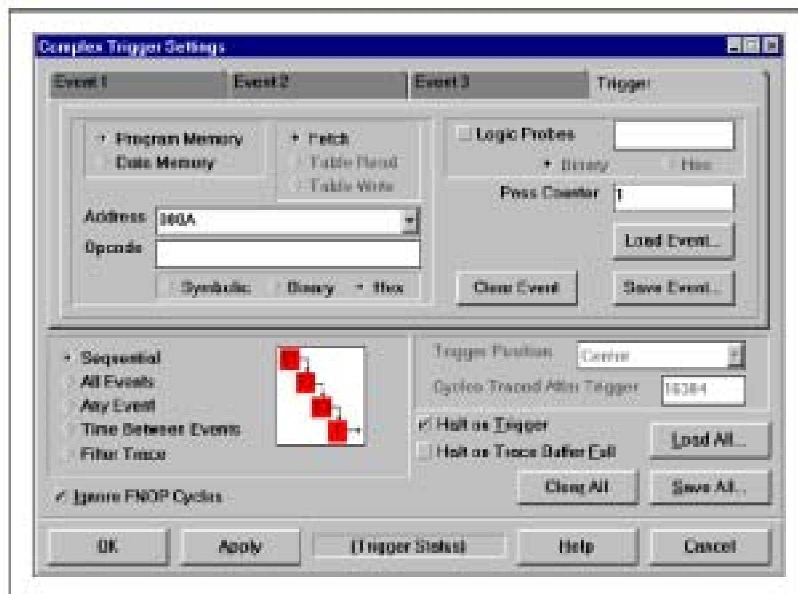


Figure 2.17: Complex Trigger Settings Dialog - Hardware Break Point

在源程序窗口中 movwf PORTB 应该改变了颜色.

通过选择 *Debug>Run>Run* 或点击工具栏绿色灯图标来运行程序.MPLAB 同样，你应该看到在 DELAY 程序的 movlw 0xFF 指令被加亮，这表示程序是在这条指令中止运行的,(有两条指令被执行过).

2.11.2 复杂触发作为一个跟踪存储器捕捉

通过选择 *Window>Trace Memory* 来打开跟踪存储器窗口.

	Addr	Op	Label	Instruction	SA	SD	DA	DD	Ex	Cycles
-8	0000	2805		goto START	—	–	—	–	11111111	0000000000
-7	0001	3FFF		(Forced NOP)	—	–	—	–	11111111	0000000001
-6	0005	0186	START	clrf 0x6	–	–	–	–	11111111	0000000002
-5	0006	1683		bsf 0x3,0x5	006	00	006	00	11111111	0000000003
-4	0007	0186		clrf 0x6	003	1E	003	3E	11111111	0000000004
-3	0008	1283		bcf 0x3,0x5	086	FF	086	00	11111111	0000000005
-2	0009	30FF	LOOP	movlw 0xFF	083	3E	083	1E	11111111	0000000006
-1	000A	0086		movwf 0x6	—	–	W	FF	11111111	0000000007
0	000B	200F		call DELAY	006	00	006	–	11111111	0000000008

Figure 2.18: Trace Memory

所有硬件断点之前所执行的指令减去一条,都被列在跟踪存储器窗口中.

> **注意:** 由于处理信号的时间,数据信息会偏移一个周期.目标数据值实际会偏移两个周期,但用于显示目的,跟踪缓存显示会补偿一个周期.

近似触发周期会被标亮为蓝色,并被编号为周期 0.其它周期的编号都会基于这个周期. 要将触发点移到显示周期的中央,选择 *Data>Find Trigger*. 要改变在跟踪存储器窗口中数据显示的方式,选择 *Options>Configure* 打开配置跟踪对话框.

关于跟踪存储器窗口的更多信息请看 6.5 节.

2.11.3 复杂触发更多功能

复杂触发可以设置为断点触发前要求最多 4 个事件发生.这些事件可以是下面三种方式组成:

- 顺序
- 所有事件
- 任何事件

另外,复杂触发可以让跟踪存储器窗口执行下面的功能:

- 事件间的定时
- 过滤跟踪

关于复杂触发的更多信息以及复杂触发的例子请看 6.3 节.

2.12 使用代码覆盖

代码覆盖特性提供了被访问(被取指,写或读)代码部分可视化的功能.代码在程序存储器窗口中用定义好的颜色加亮.颜色的定义和跟踪点文本一样,在 *Options>Environment Settings* 的 **Color** 标签中设置.

通过选择 *Debug>System Reset* 或点击工具栏上复位图标来复位程序.然后选择 *Debug>Clear All Points* 来清除所有断点,跟踪和触发点.

点击观察窗口来激活它. 然后要关闭它,可以通过点击窗口右上角的'**X**'按钮,或点击窗口左上角选择关闭,或使用 **Ctr+F4** 组合键,或选择 *File>Close*.

选择 *Windows>Program Memory* 打开程序存储器窗口.在地址 0x000A 设置一个软件断点.

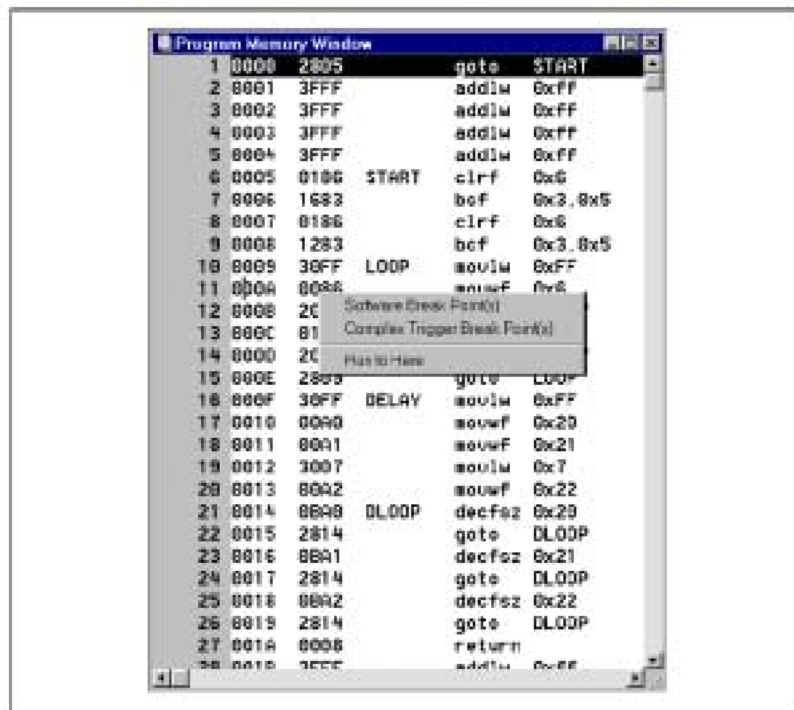


Figure 2.19: Program Memory Window

选择 _Debug>Code Coverage_ 来使能代码覆盖.在代码覆盖对话框中点击 _Enabled/Reset on Run_,然后点 **OK**.

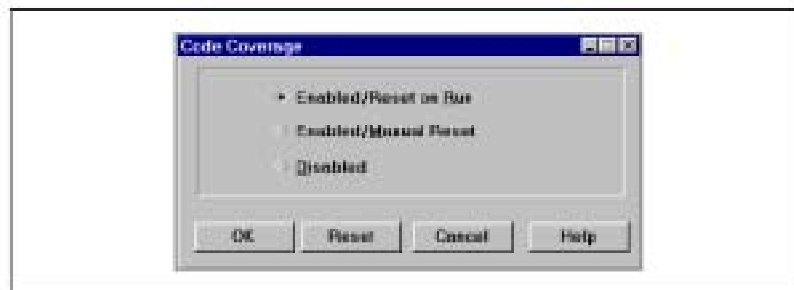


Figure 2.20: Code Coverage Dialog

选择 _Debug>Run>Run_ 或者点击工具栏的绿色灯图标来运行程序.当达到断点时,程序会中止.

所有断点之前被执行的代码,以及设断点所在的行都会被加上颜色(和跟踪点文本颜色相同).

关于代码覆盖更多信息请看 6.4 节.

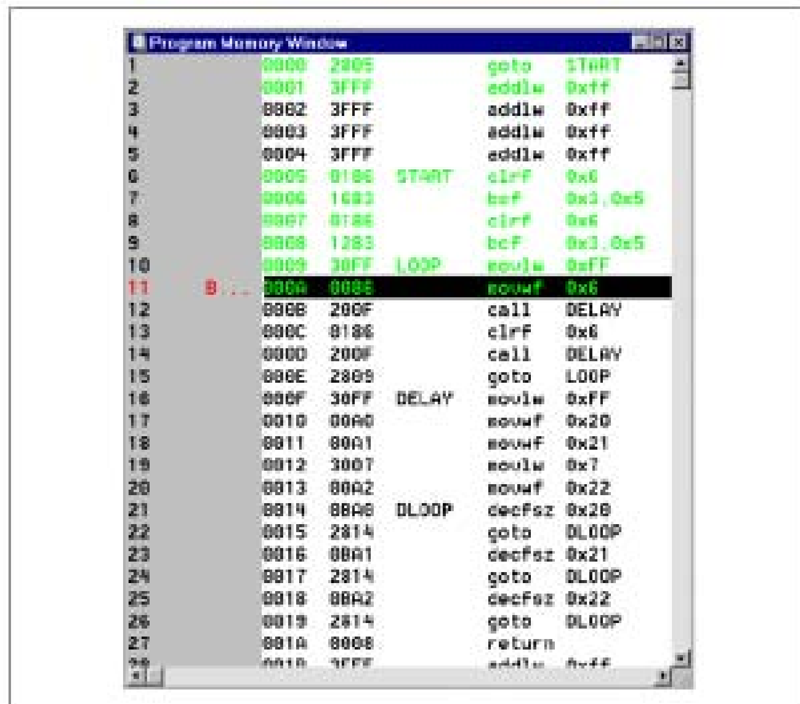


Figure 2.21: Program Memory Window - Code Coverage

2.13 总结

现在你已经完成了关于 MPLAB ICE 和 MPLAB IDE 功能的 PIC16CXXX 指南.这里演示的一个例子告诉你在使用仿真器和软件时的最基本的知识.对于更多针对具体应用的使用信息,请参考基本(第五章)和高级(第六章)特性章节.

第三章　指南-PIC18CXXX

3.1　介绍

在安装完 MPLAB ICE 硬件和 MPLAB-IDE 软件后,你就可以通过这个指南来开始工作了.

3.2　要点

这个指南包括:

- 硬件预览
- 运行MPLAB-IDE
- 设置开发模式
- 创建一个项目
- 建立项目
- 使用软件断点
- 使用命名软件断点
- 使用硬件断点
- 使用复杂触发
- 使用代码覆盖
- 总结

3.3　硬件回顾

这个指南中所使用的硬件设置见下面列表:

- **PC LPT Port:** LPT1,双向模式
- **MPLAB ICE 主机:** MPLAB ICE 2000
- **MPLAB ICE 处理器模块:** PCM18XA0

尽管你可能是使用LPT端口,而不是LPT1,但是,还是推荐你尽可能使用LPT1.而且,最好是使用双向模式.因为尽管兼容模式也能工作,但它会比较慢.

在这则指南中使用了用于 PIC18CXX2 PIC 单片机的处理器模块.其中,PIC18C252 作为一个例子. 然而其它 PIC18CXX2 处理器只需作极小的修改就能使用.

3.4 运行 MPLAB

安装完 MPLAB IDE 软件后,通过执行 MPLAB.EXE 来运行软件.

关于使用 MPLAB 的更多信息,请参考 MPLAB 用户指南和文件 README.LAB.

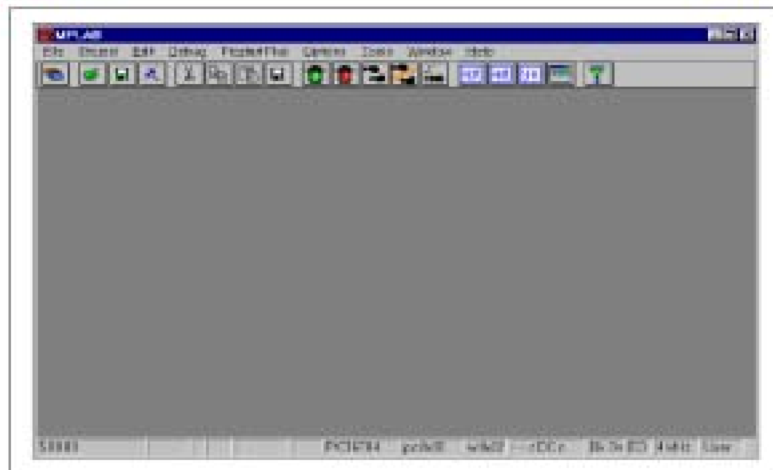


Figure 3.1: MPLAB IDE

3.5 设置开发模式

在 MPLAB IDE 软件中,打开开发模式对话框(*Options>Development Mode*)来设置 MPLAB ICE 仿真器.通过点击对话框的每个标签和下面的设置选项来设置开发模式.

3.5.1 端口标签

端口标签显示主机 PC 上空闲 LPT 端口的信息.

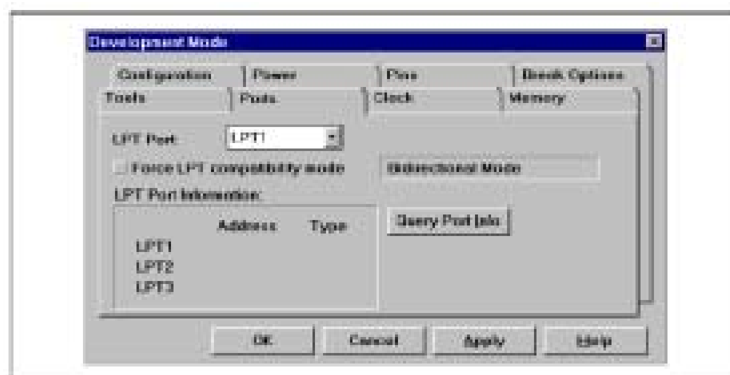


Figure 3.2: Development Mode Dialog – Ports Tab

从下拉列表中选择可供使用的 LPT 端口.要决定这个 LPT 端口是如何配置到你的系统,点击 **Query Port Info**.推荐 MPLAB ICE 的运行模式是双向模式(PS/2, EPP 和 ECP 类型).

点击 **Apply** 接受这个设置.

如果你还有任何关于 LPT 端口配置的问题,请参考设置章节或常见问题的细节.

3.5.2 工具标签

工具标签显示了开发模式和器件信息.

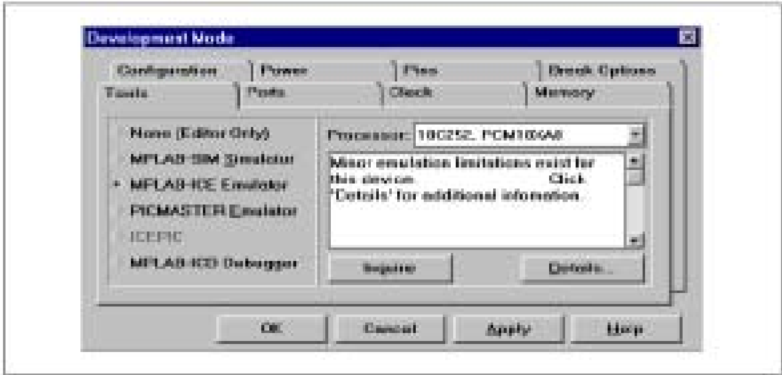


Figure 3.3: Development Mode Dialog – Tools Tab

在开发模式中选择 MPLAB ICE 仿真器

WARNING	如果有其它的设备连在并行口上,千万不要选择 MPLAB ICE 仿真模式,否则会对那个设备造成永久性的损害.

从下拉列表中选择 PIC18C252 来仿真.在处理器的下面,是关于 MPLAB ICE 仿真器仿真这款型号时的一些限制的简单描述.更多细节可以点击细节进行查看.

点击应用接受这个设置.

3.5.3 中断选项标签

中断选项标签用来改变全局中断和跟踪点环境选项.默认值为下载清除断点,全局中断使能和冻结外设.在这则指南中,不检查中止冻结外设.

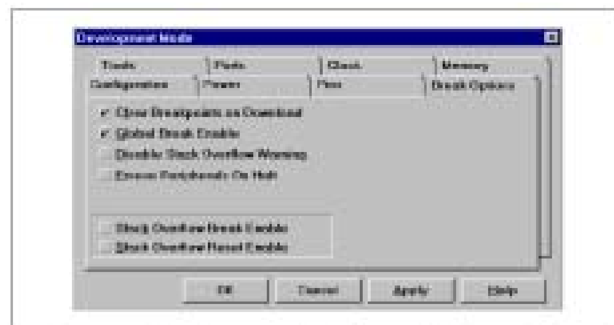


Figure 3.4: Development Mode Dialog – Break Options Tab

点击 **Apply** 接受设置.

3.5.4 其它标签

其它标签不需要再编辑.然而,你也许希望查看它们,使你熟悉那些选项.

- **电源**标签: 显示系统电源信息. 默认是处理器从仿真器供电.
- **时钟**标签: 显示振荡器类型和频率信息. 默认是使用板载时钟,振荡器类型为 LP,频率为 4.0MHz.
- **存储器**标签: 使用这个标签来设置被使用的存储器配置.对于选定的处理器,这个标签不能使用.
- **配置**标签: 使用这个标签来设置看门狗定时器和/或处理器模式.默认是看门狗定时器关闭.对于选定的处理器,处理器模式不可用.
- **引脚**标签: 使用这个标签来设置引脚,如使能/关闭主清除(MCLR).对于选定的处理器,这个选项不可用.

当你设置完毕,点击 **OK** 接受设置,并关闭开发模式对话框.

3.6 创建一个项目

使用 MPLAB IDE 软件开发一个项目最好的方法是创建一个项目.在这个指南中,你将要创建一个项目,名称为 icetut18.pjt.然而在这之前,你应该先建一个名字为 icetut18 的文件夹.你建的项目将放在这个文件夹中.同时,你应该将 MPLAB 安装目录下的 icetut18.asm 文件拷贝到这个文件夹中.

通过选择 *Project>New Project* 来创建一个新项目.新项目对话框会出现.

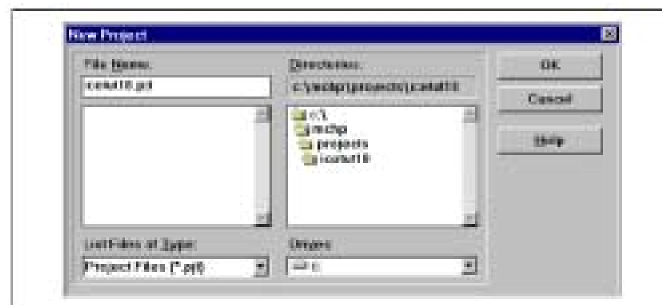


Figure 3.5: New Project Dialog

找到你创建项目的目录,然后给这个项目命名为 icetut18.pjt.点击 **OK** 关闭对话框并打开编辑项目对话框.

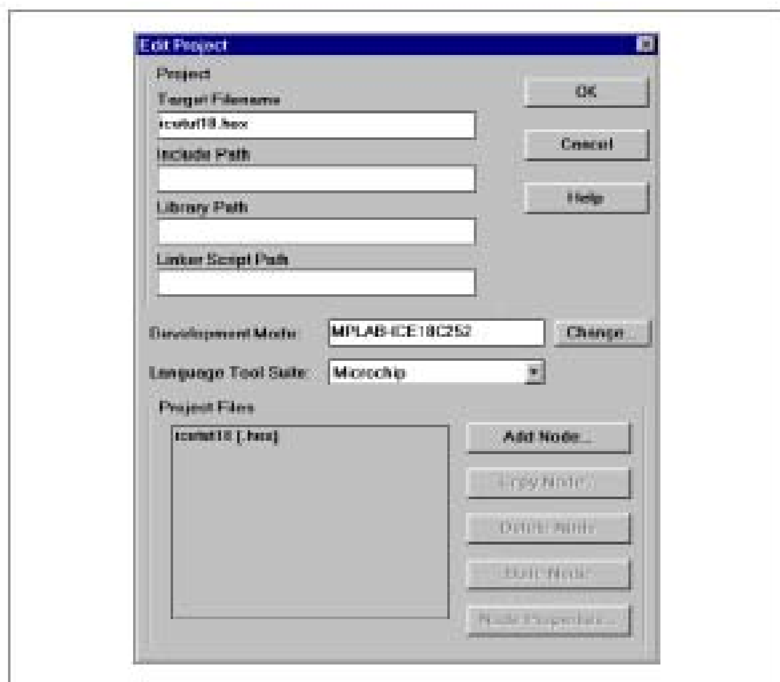


Figure 3.6: Edit Project Dialog – Hex File Only

在项目文件对话框中列出的一个文件是 `icetut18[.hex]`, 这个文件将作为输出文件.

> **注意:** 如果你使用的 MPLAB IDE 不是最近才装的,(即,默认配置可能已经更改),你将需要:
> - 点击Hex文件来激活节点属性按钮
> - 点击这个按钮打开接到属性
> - 在这个对话框中选择MPASM作为语言工具
> - 点击**OK**关闭对话框并返回编辑项目对话框

紧挨着项目文件框是一组按钮,其中一个**添加节点**按钮处于激活状态.点击它打开添加节点对话框.

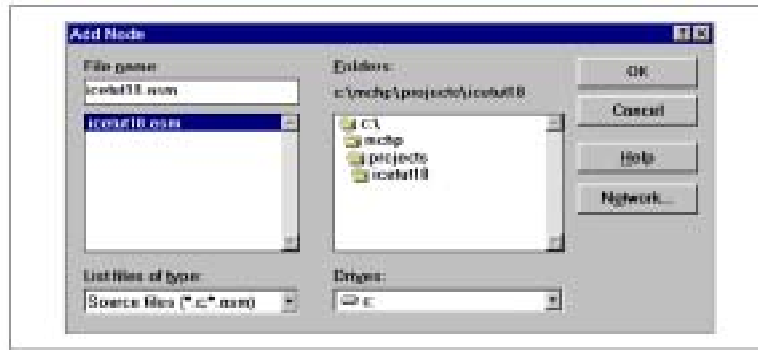


Figure 3.7: Add Node Dialog

找到你拷贝到项目目录中的 ASM 文件,然后点击它的名字选择它.如果你找不到这个文件,或者如果它丢失了,在对话框的文件名这一栏中输入 icetut18.asm.

点击 **OK** 关闭对话框并返回到编辑项目对话框.



Figure 3.8: Edit Project Dialog – Hex and ASM Files

现在 ASM 文件 icetut18[.asm]应该列出在 Hex 文件 icetut18[.hex]下面.点击 **OK** 关闭对话框.

如果你能找到并拷贝文件 icetut18.asm 到项目目录中,你现在应该在 MPALB IDE 主窗口中,而没有其它窗口打开.这样可以进行下一步.

如果你不能找到这个文件,你现在应该看到 MPLAB IDE 主窗口中有一个打开的文件窗口,名字为 Untitled.

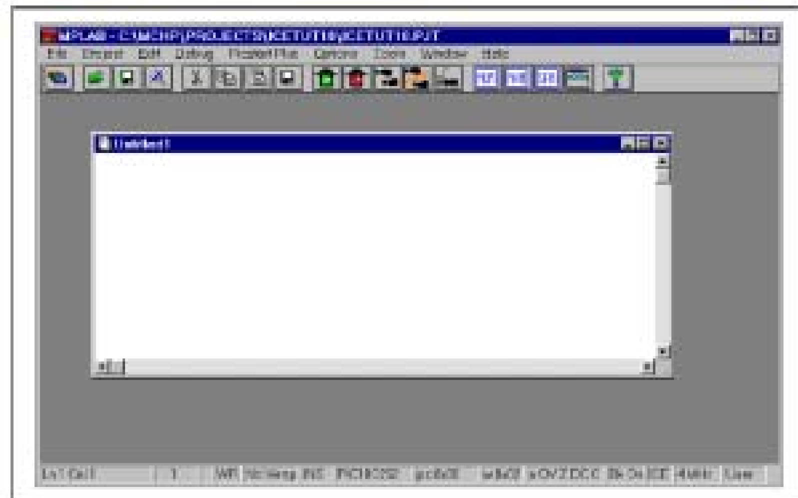


Figure 3.9: MPLAB IDE with Untitled File Window

你可以通过 *File>Save As* 给这个文件窗口命个名.当 Save File As 对话框打开时,选择这个项目目录 `icetut18`,并输入 `icetut18.asm` 作为文件名.点击 **OK**. 现在你就给这个文件命了名,并准备开始在这个空窗口中输入源代码.

icetut18.asm 源代码：

```
        list    p=18c252
        title   "PIC18C252 Tutorial - Flash PORTB LEDs"

        #include <p18c252.inc>

TEMP1   equu    0x00            ;Delay loop
TEMP2   equ     0x01            ;temp. variables
TEMP3   equ     0x02

        org     0x00            ;Reset Vector
        goto    START

        org     0x20            ;Start Program
START
        clrf    PORTB           ;Clear PortB (LED's off)
        clrf    TRISB           ;Set PortB as output
LOOP
        movlw   0xFF
```

```
        movwf   PORTB               ;Set PortB
        call    DELAY               ;Wait
        clrf    PORTB               ;Clear PortB
        call    DELAY               ;Wait

        goto    LOOP                ;Repeat
;*******************************************
;*     This routine is a software delay.    *
;*     Fosc = 1/Tosc; Tcycle = 4 x Tosc     *
;*     Delay = TEMP1xTEMP2xTEMP3xTcycle *
;*******************************************

DELAY
        movlw   0xFF
        movwf   TEMP1               ;TEMP1 = 255
        movwf   TEMP2               ;TEMP2 = 255
        movlw   0x07
        movwf   TEMP3               ;TEMP3 = 7
DLOOP
        decfsz  TEMP1, F
        goto    DLOOP

        decfsz  TEMP2, F
        goto    DLOOP

        decfsz  TEMP3, F
        goto    DLOOP

        return

        END
```

通过 File>Save 保存这个文件.然后通过 File>Close 或点击窗口右上角关闭文件.

3.7 建立这个项目

在这个指南中,建立项目也就是编译源代码.因为在项目中只有一个 ASM 文件.选择 *Project>Build All* 来建立这个项目.当完成时,建立结果窗口会出现.

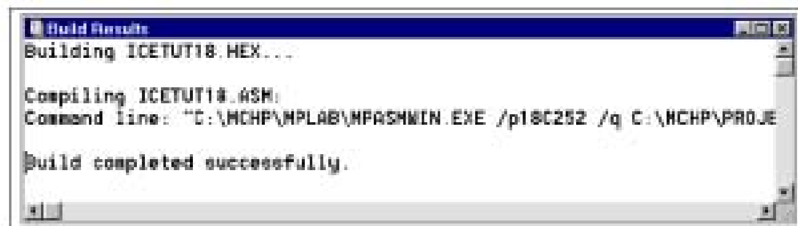


Figure 3.10: Build Results Window

如果你是自己输入的源代码,且建立失败,请检查你的拼写然后再尝试建立.

更多信息请参考 MPLAB 用户指南.

3.8 使用软件断点

现在你的项目已被建立好,并且代码成功编译成一个可执行(.hex)程序.你可以在 MPLAB ICE 上运行这个程序.

接着打开源代码文件.首先选择 *File>Open* 打开 Open Existing File 对话框.在目录列表中找到这个项目目录,选择 `icetut18.asm`. 点击 **OK**.

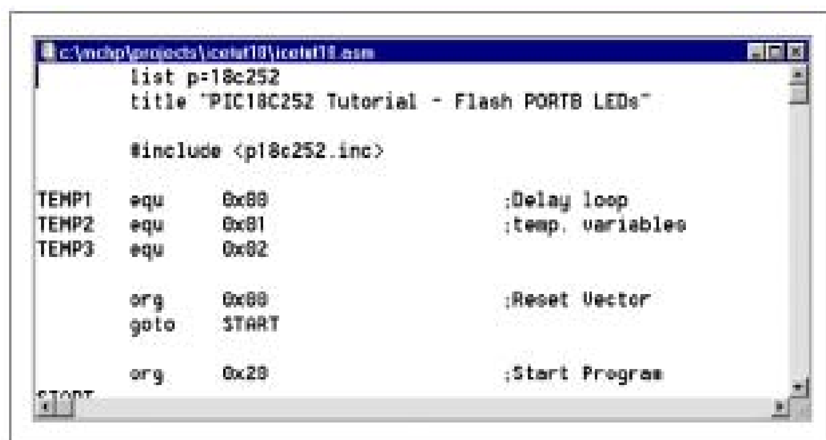


Figure 3.11: Source Code File Window

接着,通过选择 *Window>Watch Windows>New Watch Window* 打开一个新的观察窗口. 在添加观察窗口对话框滚动列表中找到 PORTB, 点击选中 PORTB 选项.点击 **Add**. PORTB 应该出现在观察窗口中.点击 **Close** 关闭添加观察窗口对话框.

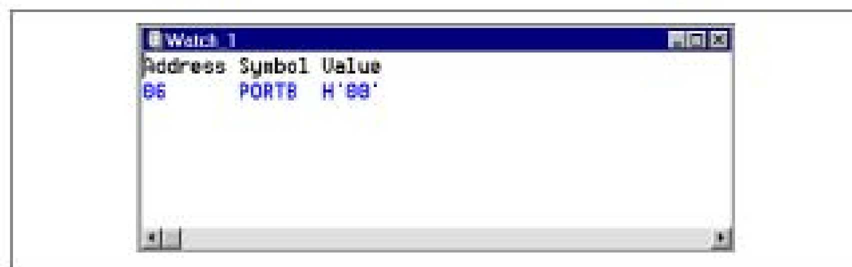


Figure 3.12: Watch Window

选择 *Debug>System Reset* 或点击工具栏的复位图标复位程序.

现在你可以在源代码文件窗口中设置一个软件断点.回到程序的主循环,在 movwf PORTB 这一行点击鼠标右键.会出现一个菜单.选择软件断点.这一行会改变颜色.如果你不喜欢这种颜色,你可以通过选择 *Options>Environment Setup* 来改变.点击颜色标签,选择一种不同的颜色用于断点文本.

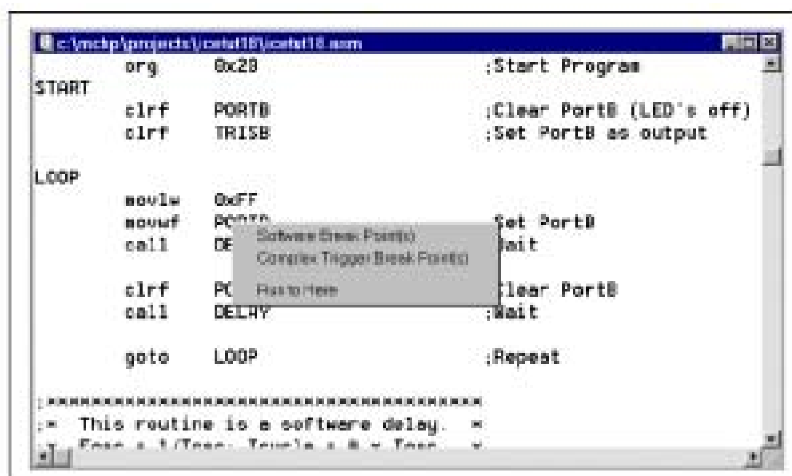


Figure 3.13: Set Software Break Point

通过 *Debug>Run>Run* 或点击工具栏上的绿色灯图标来运行程序. MPLAB IDE 窗口的底端状态栏会改变颜色,显示程序在运行.当程序达到断点时会中止.

通过选择 *Debug>Run>Step*,或按**<F7>**,或点击工具栏上的单步图标来执行单步.你应该看到观察窗口中 PORTB 的值已经从 H' 00 ' 变为 H' FF ',并且应该改变了颜色.

这演示了软件断点是如何在 MPLAB ICE 中工作的.在这行代码被执行之前,断点会终止仿真运行.然后单步执行一次,代码会被执行并显示在观察窗口中.

你也可以从鼠标右键菜单中选择 Run to Here,而不是 Software Break Point,来设置一个临时断点.

3.9 使用命名软件断点

MPLAB 允许最多 16 个命名软件断点.这些断点可以被选择性的使能和关闭.以这种方式设置的断点可以随着项目一

起保存.鼠标右键菜单断点不可以.

通过选择 *Debug>System Reset* 或点击工具栏上复位图标来复位程序.这会清除所有断点.

通过选择 *Debug>Break_Settings* 打开断点设置对话框. 在对话框的开始栏上输入 0x0026 这样将 **break1** 断点指定在指令 movwf PORTB,然后点击 **Add**.

> **注意:** 要决定一条指令的地址,可以使用程序存储器窗口(*Window>Program Memory*).

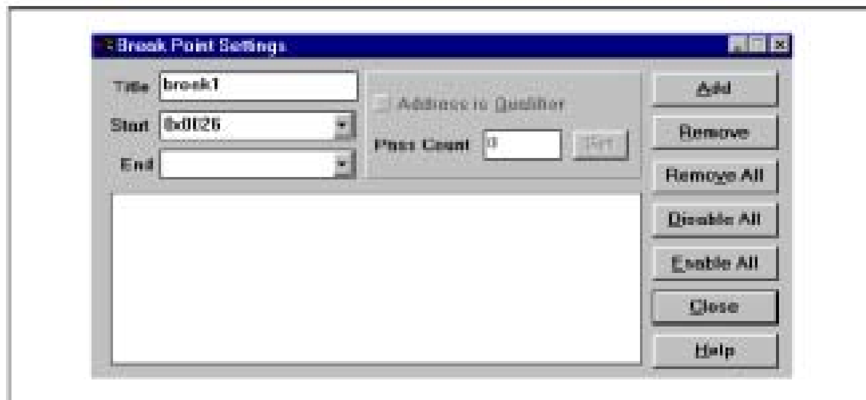


Figure 3.14: Break Point Settings Dialog

你应该在断点设置对话框对应0x0026的 **break1** 处于使能状态,在源程序窗口相应指令行应改变颜色. 点击断点设置对话框上的 **Close** 关闭窗口.

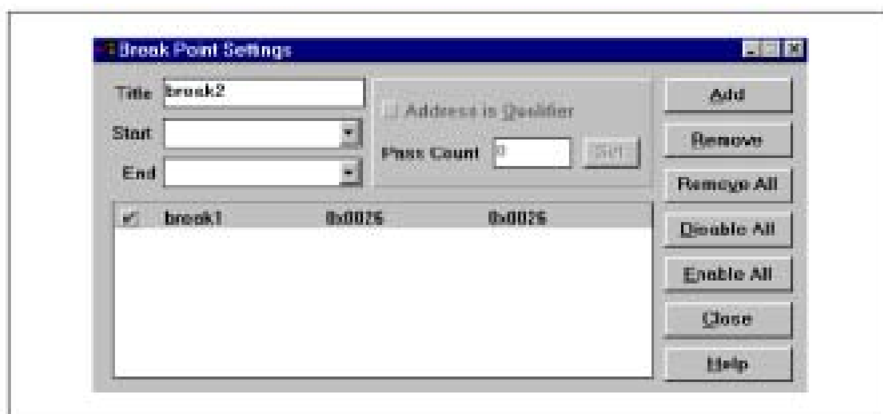


Figure 3.15: Break Point break1 Set

通过选择 *Debug>Run>Run* 或点击工具栏绿色灯图标来运行程序.MPLAB IDE 窗口底端的状态栏会改变颜色,表示程序正在运行.当程序运行到断点会中止.

单步运行,你可以看到观察窗口中的 PORTB 值从 H'00'变为 H'FF'.

3.10 使用硬件断点

除了在程序存储器地址设置软件断点外,也可以设置硬件断点.

通过选择 *Debug>System Reset* 或点击工具栏上复位图标来复位程序.这会清除所有断点.

跳到程序的主循环,在 movwf PORTB 上点击鼠标右键. 会出现一个菜单.这时选择复杂触发断点.这一行会改变颜色,如果你不喜欢这种颜色,你可以选择 *Options>Environment Setup*,点击 Color 标签,然后选择一种你喜欢的颜色.

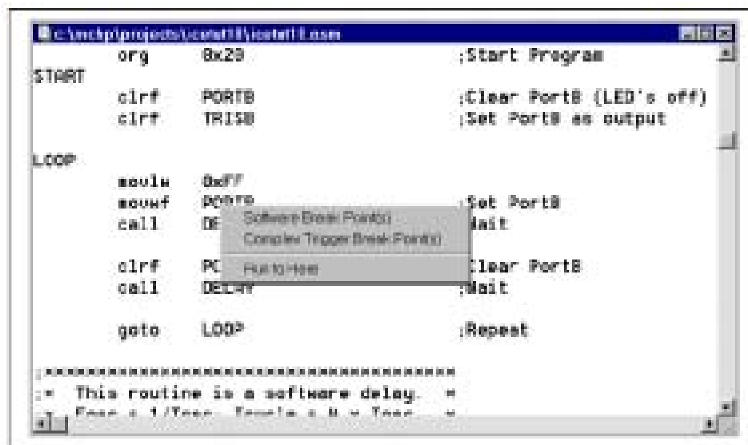


Figure 3.16: Set Hardware Break Point

通过选择 *Debug>Run>Run* 或点击工具栏绿色灯图标来运行程序.MPLAB IDE 窗口底端的状态栏会改变颜色,表示程序正在运行.当程序中止.它会回到原来的颜色.

注意,程序并不在你设断点的地方停止.你应该看到在 DELAY 程序的 movlw 0xFF 指令被加亮,这表示程序是在这条指令中止运行的.

这表示,当你使用硬件断点来中止运行时,在处理器被中止前,你可能会跳过一条或多条额外的指令.

3.11 使用复杂触发

设置复杂触发设置对话框可以完成:

- 一个硬件断点
- 一个跟踪存储器捕捉

当使用了这个对话框,这个项目的复杂触发设置就被保留了. 鼠标右键菜单复杂触发断点不可用.

在这个指南中,复杂触发会首先用作一个硬件断点,然后作为一个跟踪存储器捕捉.

3.11.1 复杂触发作为一个硬件断点

通过选择 *Debug>System Reset* 或点击工具栏上复位图标来复位程序.这会清除所有断点.

要设置复杂触发,可以通过选择 *Debug>Complex Trigger Settings* 打开复杂触发设置对话框.设置复杂触发 **Ignore FNOP Cycles** 和 **Halt on Triigger,**这样它就可以象鼠标右键复杂触发断点一样工作了(如,在地址 0x0026 作为一个单事件顺序触发).

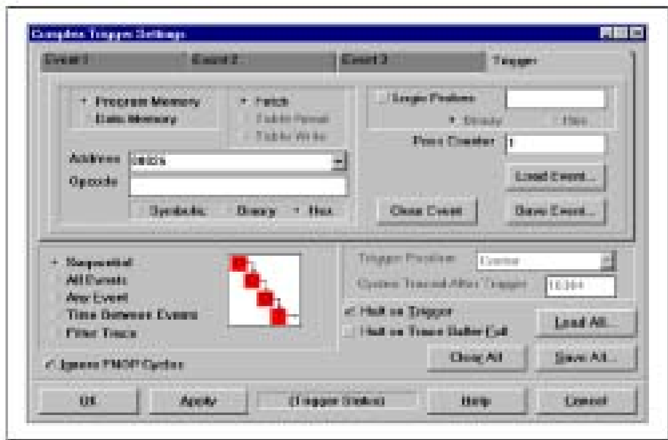


Figure 3.17: Complex Trigger Settings Dialog - Hardware Break Point

在源程序窗口中 movwf PORTB 应该改变了颜色.

通过选择 *Debug>Run>Run* 或点击工具栏绿色灯图标来运行程序.MPLAB 同样, 你应该看到在 DELAY 程序的 movlw 0xFF 指令被加亮, 这表示程序是在这条指令中止运行的,(有两条指令被执行过).

3.11.2 复杂触发作为一个跟踪存储器捕捉

通过选择 *Window>Trace Memory* 来打开跟踪存储器窗口.

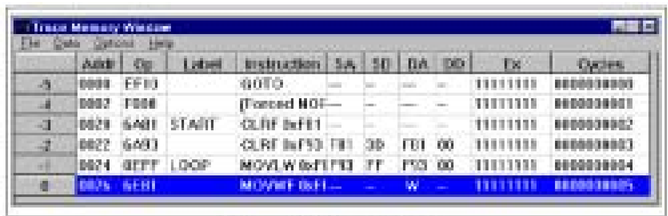


Figure 3.18: Trace Memory

所有硬件断点之前所执行的指令减去一条,都被列在跟踪存储器窗口中.

> **注意:** 由于处理信号的时间,数据信息会偏移一个周期.目标数据值实际会偏移两个周期,但用于显示目的,跟踪缓存显示会补偿一个周期.

近似触发周期会被标亮为蓝色,并被编号为周期 0.其它周期的编号都会基于这个周期. 要将触发点移到显示周期的中央,选择 *Data>Find Trigger*. 要改变在跟踪存储器窗口中数据显示的方式,选择 *Options>Configure* 打开配置跟踪对话框.

关于跟踪存储器窗口的更多信息请看 6.5 节.

3.11.3 复杂触发更多功能

复杂触发可以设置为断点触发前要求最多 4 个事件发生.这些事件可以是下面三种方式组成:

- 顺序

- 所有事件
- 任何事件

另外,复杂触发可以让跟踪存储器窗口执行下面的功能:

- 事件间的定时
- 过滤跟踪

关于复杂触发的更多信息以及复杂触发的例子请看 6.3 节.

3.12 使用代码覆盖

代码覆盖特性提供了被访问(被取指,写或读)代码部分可视化的功能.代码在程序存储器窗口中用定义好的颜色加亮.颜色的定义和跟踪点文本一样,在 *Options>Environment Settings* 的 **Color** 标签中设置.

通过选择 *Debug>System_Reset* 或点击工具栏上复位图标来复位程序.然后选择 *Debug>Clear_All_Points* 来清除所有断点,跟踪和触发点.

点击观察窗口来激活它. 然后要关闭它,可以通过点击窗口右上角的'**X**'按钮,或点击窗口左上角选择关闭,或使用 **Ctr+F4** 组合键,或选择 *File>Close*.

选择 *Windows>Program Memory* 打开程序存储器窗口.在地址 0x000A 设置一个软件断点.

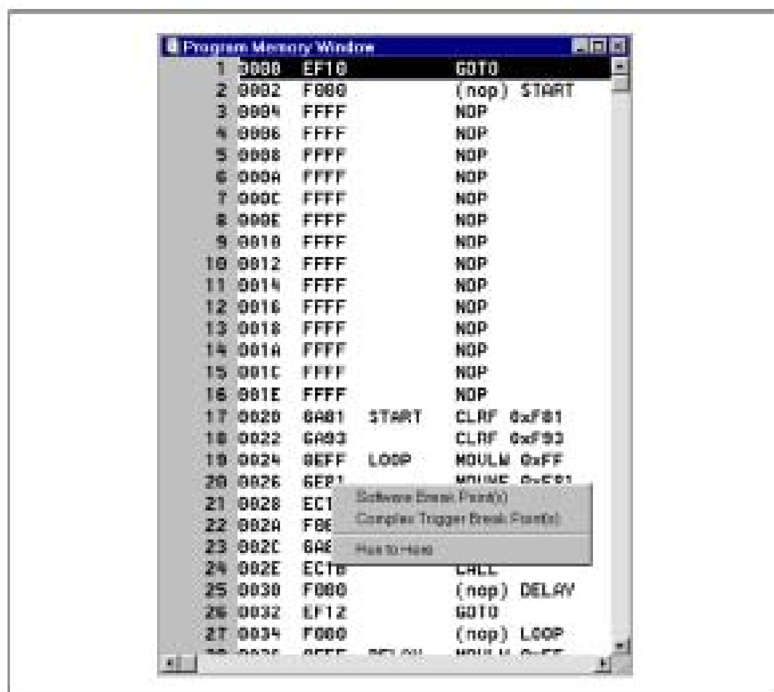


Figure 3.19: Program Memory Window

选择 *Debug>Code Coverage* 来使能代码覆盖.在代码覆盖对话框中点击 *Enabled/Reset on Run*,然后点 **OK**.



Figure 3.20: Code Coverage Dialog

选择 Debug>Run>Run 或者点击工具栏的绿色灯图标来运行程序.当达到断点时,程序会中止.所有断点之前被执行的代码,以及设断点所在的行都会被加上颜色(和跟踪点文本颜色相同).关于代码覆盖更多信息请看 6.4 节.

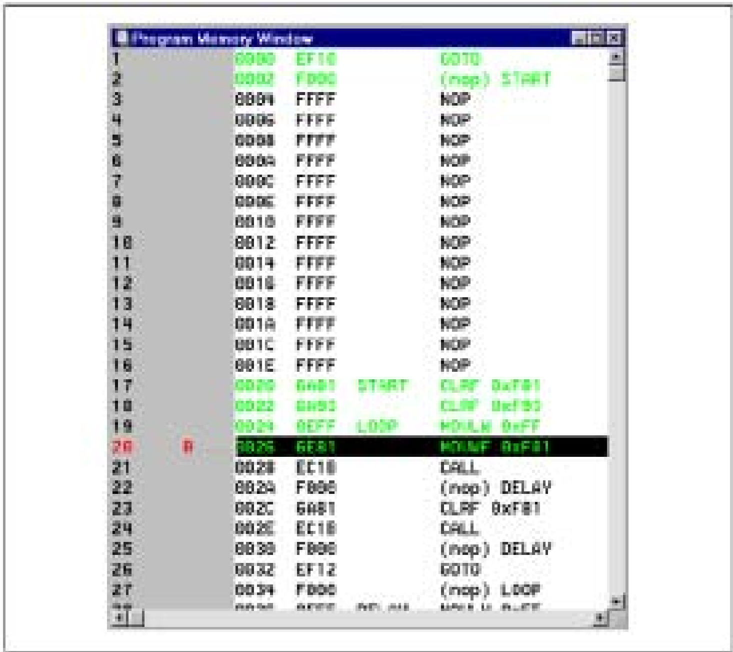


Figure 3.21: Program Memory Window - Code Coverage

3.13 总结

现在你已经完成了关于 MPLAB ICE 和 MPLAB IDE 功能的 PIC18CXXX 指南.这里演示的一个例子告诉你在使用仿真器和软件时的最基本的知识.对于更多针对具体应用的使用信息,请参考基本(第五章)和高级(第六章)特性章节.

第四章　通用设置

4.1　介绍

在安装并运行 MPLAB 软件后,MPLAB ICE 必须被使能并指定正确的仿真芯片型号.

4.2　要点

使用 MPLAB ICE 的步骤为:

- 运行MPLAB
- 配置LPT 端口
- 选择开发模式和处理器
- 选择处理器电源
- 设置处理器时钟
- 设置各种硬件
- 使用MPLAB项目

4.3　运行 MPLAB

安装完 MPLAB 后,通过执行 MPLAB.EXE 来运行软件.

更多安装和使用 MPLAB 的信息,请参考 MPLAB 用户指南和包含文件 README.LAB.



Figure 4.1: MPLAB IDE

4.4 配置 LPT 端口

选择 *Option>Development Mode* 来显示 PC 并口的信息.这个对话框的端口标签显示主机 PC 上可用 LPT 端口的信息. 从下拉列表中选择可供使用的 LPT 端口.要决定这个 LPT 端口是如何配置到你的系统,点击 **Query Port Info**.

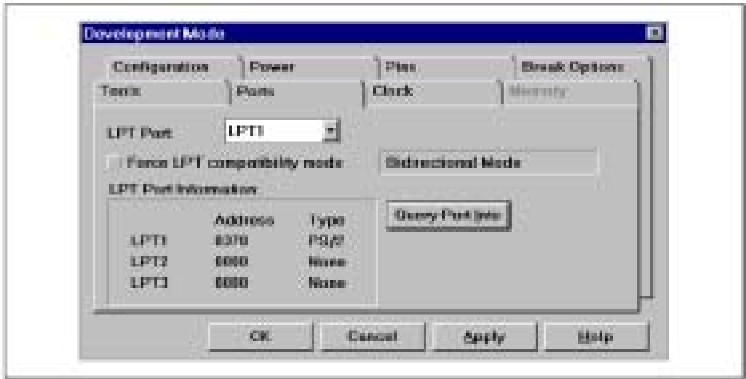


Figure 4.2: Development Mode Dialog – Ports Tab

根据你的 PC 的主板,最多可以有四种不同的并口:

- SPP
- PS/2
- EPP
- ECP

SPP 口是原始并行口.它设计用于发送 8 位宽的数据,但并没有设计用于数据的接收.数据可以通过使用状态信号来接收,但是一次只能接收 4 位.这种操作模式被称为可兼容模式.

PS/2,EPP,和 ECP 类型的并行口可以进行双向的通讯.也就是说,它们可以发送和接收 8 位宽的数据.绝大多数新的 PC 包含一个双向口,或者一个端口既可以配置为双向口也可以配置为可兼容模式.

如果 LPT 端口支持的话,MPLAB ICE 是以双向模式通讯的.然而,如果主 PC 通过 LPT 端口通讯有困难的话,可以尝试以兼容模式进行通讯.要迫使兼容模式通讯,可选择 **Force Compatibility Mode**. 同时,要检查 PC 主板上的 LPT 端口,可以查看 BIOS 并行口设置,或查看用于并行口的跳线.

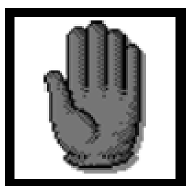
注意: 兼容模式通讯会比双向模式慢.推荐只有在双向模式不能通讯的情况下才使用兼容模式.

如果通讯仍有问题,请参考第八章.

点击应用接受设置.

4.5 选择开发模式和处理器

要选择 MPLAB ICE 仿真器作为开发模式,选择开发模式对话框的**工具**标签并点击 MPLAB ICE Emulator.

WARNING	
	如果有任何其它的设备安装在并口上,请千万不要选择 MPLAB ICE Emulator 模式,否则,可能会对那个设备造成永久性的损害.

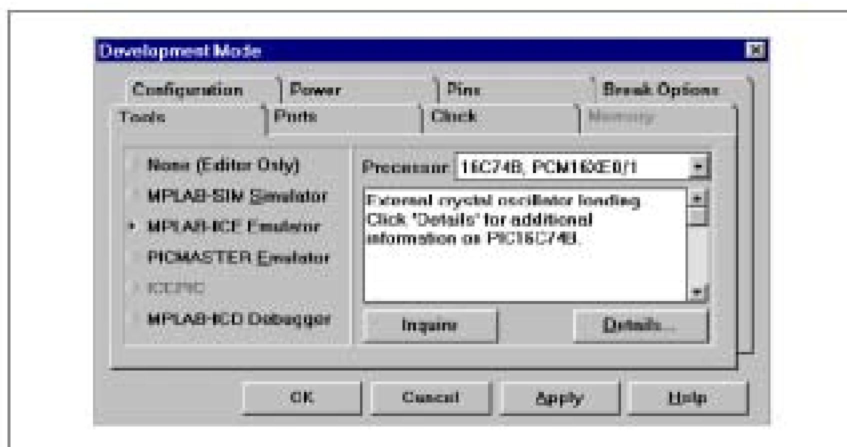


Figure 4.3: Development Mode Dialog – Tools Tab

当选择好 MPLAB ICE Emulator 后,需要确定处理器模块最终仿真哪款型号的单片机.由于绝大多数处理器模块都能仿真几个不同的型号,所有型号可能需要改变.可供选择的处理器列表在 Processor 下列列表中显示. 选择期望的型号.

处理器下面是一个简单的描述. 介绍了使用 MPLAB ICE 仿真时的一些限制.更多的细节可以点击细节查看.

4.6 选择处理器电源

MPLAB ICE 运行仿真处理器芯片由仿真器主机(5V)供电,或由目标系统(2.0-5.5)供电. 当第一次初始化时,仿真器默认是通过仿真器供电.

4.6.1 通过仿真器(系统电源)供电

选择开发模式对话框的电源标签.在处理器电源下面,选择通过仿真器,点击应用.

> **注意:** 仿真系统不能通过器件适配器给目标板供电.如果器件适配器插进一块无电源的目标板,由于通过器件适配器 Vcc 的漏电流,目标板 的 Vcc 在 1-3V 的电压级别是不常见的. 当目标板还没有供电时,MPLAB 的初始化也会有问题.

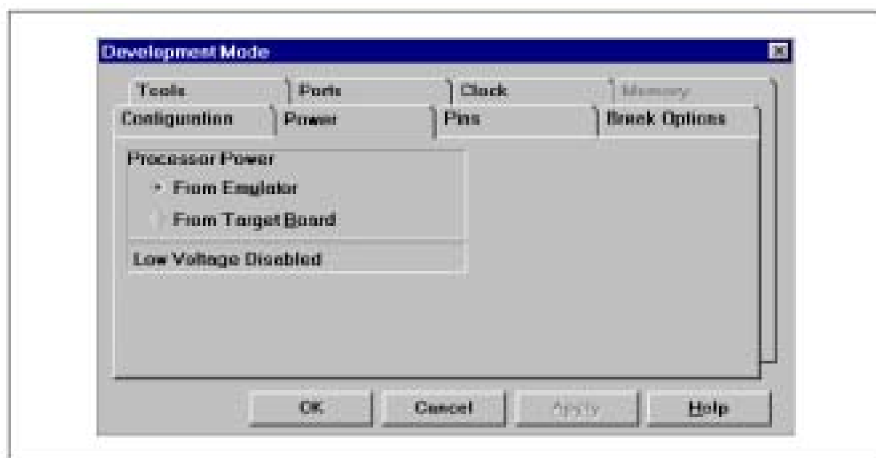


Figure 4.4: Development Mode Dialog – Power Tab, From Emualtor

4.6.2 从目标板供电

选择开发模式对话框的电源标签.在处理器电源下面,选择通过目标板,点击应用.

在这样配置之前,请先参考 MPLAB ICE 处理器模块和器件适配器电气特性(DS51140).

> **注意:** 如果你使用从目标板供电,那么确保目标板供电正常,否则仿真器会工作不正常.如果器件适配器插进一块为供电的目标板,仍然会有一些漏电流通过器件适配器的 Vcc. 当目标板未供电时,MPLAB 也将无法初始化.

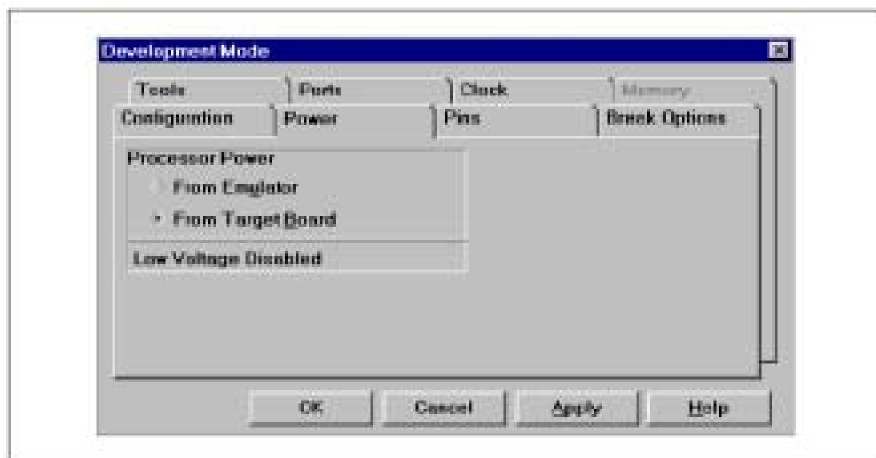


Figure 4.5: Development Mode Dialog – Power Tab, From Target Board

4.6.3 低电压仿真

MPLAB ICE 也支持低电压仿真,从 2.0V 至 4.6V.

仿真器系统不能提供除了 5V 以外的任何级别的电压给仿真处理器. 在这个配置中,电源必须通过目标板供电.

> **注意:** 在将仿真系统连接到目标应用系统前,打开仿真系统电源(处理器模块已经插入),选择处理器电源来自仿真器以运行 MPLAB 执行初始化. 然后选择电源来自目标板,连接目标板到仿真系统,给目标板加电.

4.7 设置处理器时钟

MPLAB ICE 既可以使用仿真器自带时钟,也可以使用目标板时钟.

4.7.1 使用板载时钟

MPLAB ICE 有一板载时钟,它可以被编程为 32kHz 至 40MHz.要编程时钟,选择开发模式中的**时钟**标签.

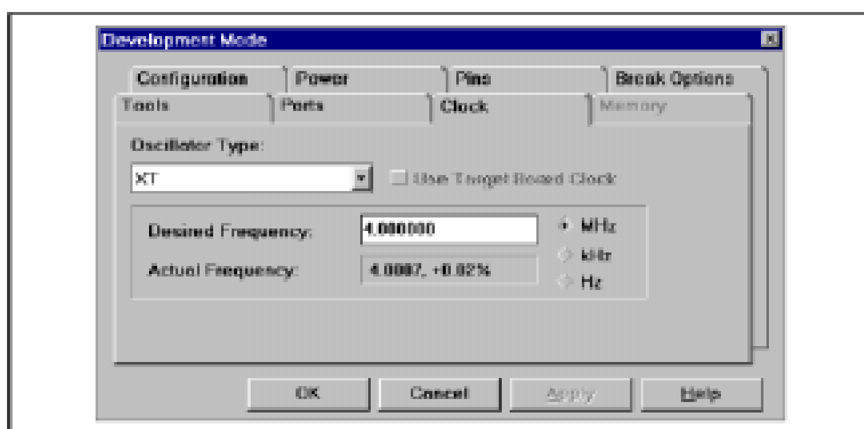


Figure 4.6: Development Mode Dialog – Clock Tab, On-Board Clock

首先,根据期望的频率范围选择振荡器类型.对于每种振荡器类型,参考相应器件的数据手册来决定支持的频率范围.

然后,选择期望频率的数量级(MHz, kHz, 或 Hz),输入期望频率,点击 **Apply**.接着时钟就会编程为尽可能按照那个频率来运行.由于产生的时钟频率会与期望的时钟频率有轻微的差别,所有实际的频率会显示出来.实际频率与期望频率的差异会在 0.5%以内.

要验证时钟频率,你可以测出 TRIGOUT 逻辑探针的触发输出脉冲然后除以 4.

4.7.2 使用目标板时钟

只要目标板电源在使用,MPLAB ICE 就能使用目标板上的处理器时钟它决定目标板时钟频率(小于 3.5%),并能使用它用于显示定时信息.

要使用目标板时钟并决定它的频率,首先选择开发模式对话框的电源标签,设置处理器电源来自目标板.然后选择时钟标签并选择使用目标板时钟.

注意: 如果 MPLAB ICE 没有连接到目标板,你点击使用目标板时钟,你会得到一个警告对话框.

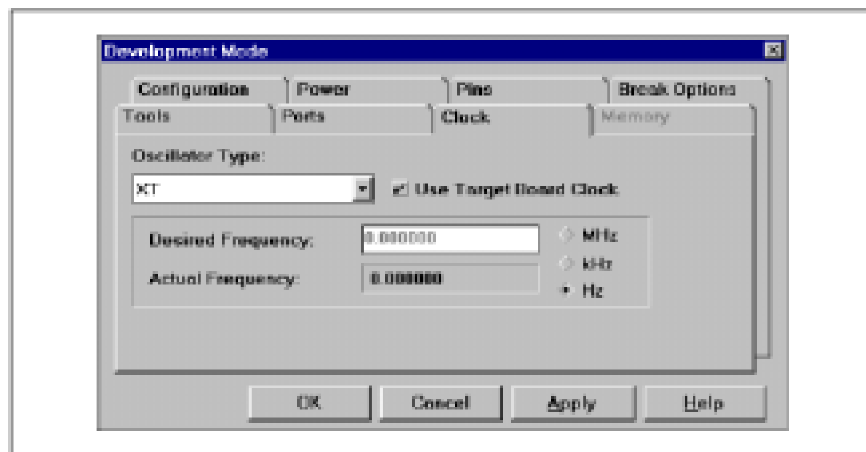


Figure 4.7: Development Mode Dialog – Clock Tab, Target Clock

目标板时钟频率会被计算出,显示出来,并用于任何内部时间的计算.如果频率超出了 32kHz 至 40MHz 这个允许的范围,就会发出一条警告.

由于测量错误,计算出来的频率可能不是内部时间计算所期望的值.(如,你的晶体振荡器频率为8MHz±50ppm,但是目标频率显示为7.993755.)频率可以通过在期望频率栏中键入期望的值来改变.

点击应用接受这个设置.

4.8 设置相关硬件

除了你已经完成的设置,还有一些设置你可能会改变,也可能不改变.对于这些设置更详细的信息,请参考 MPLAB 用户指南.

当你完成所有期望的设置后,点击 **OK** 保持这些设置并关闭开发模式对话框.

4.8.1 存储器标签

使用存储器标签来设置被使用的存储器配置.这个选项并不是对所有型号都可用的.

对于存储器映射的外围范围,指定开始和结束高字节地址.可以输入的用于高位字节的最大值是 0xFE.

为了能使用片外存储器,处理器模式必须设为微处理或扩展微处理器.

Disable Long Writes 用于关闭/使能长写. 如果是对内部存储器进行表写,你应该选择关闭长写.这些表写指令应该在器件编程前被移走.

4.8.2 配置标签

看门狗定时器设置

决定是否会需要看门狗定时器(WDT).在开发期间,WDT 通常被关闭.然而,如果一个特定的开发模式需要使能 WDT,要记得设它的预分频.

处理器模式设置

如果处理器支持外部存储器模式(微处理器或扩展微处理器),选择外部存储器是由目标板还是仿真器提供.同时声明任何存储器映射外围的开始和结束地址.

4.8.3 引脚标签

设置处理器引脚,如 MCLR.

4.8.4 中断选项标签

使用中断选项标签来改变全局中断和跟踪点环境选项.这些包括下载清除断点,全局断点使能,关闭堆栈溢出警告和中止外设冻结.其它选项为堆栈溢出中断使能和堆栈溢出复位使能.

4.9 使用 MPLAB 项目

4.9.1 创建一个项目

在 MPLAB 环境里开发和调试代码是基于项目的.尽管仿真运行可以不用打开一个项目,不过项目有下面一些优点:

- 单个或多个源文件可以轻松的建立和维护
- 可使用符合调试
- 调试环境可以保存,便于以后使用

更多信息请参考MPLAB用户指南

4.9.2 保存项目

一旦你将 MPLAB 系统和软件配置好,建议你保存当前的项目.这样可以保存设置方便以后使用,或在你继续调试时作为一个备份或默认值.

开发模式

选中的开发模式会随着项目一起保存.要改变开发模式,可以按照下面的步骤:

- 打开项目,以前使用过的开发模式会被选择.
- 选择 *Options>Development Mode* 访问开发模式对话框,然后改变开发模式
- 保存项目

处理器时钟

时钟源和频率会随着项目信息一起保存.如果使用仿真器可编程时钟,当项目重新打开时,它会设到相同的频率.然而,如果使用目标板时钟,那么下次打开项目时,时钟有可能会不工作.当项目重新打开时,系统会试图使用目标板时钟,并决定它的频率.如果目标板时钟无法工作,会发出一个警告,并使用 4MHz 系统时钟.所以有必要保证仿真器和处理器模块初始化正确.

4.9.3 打开/关闭一个项目

打开以前创建好的项目,选择 *Project>Open Project*.然后选择项目文件(*.PJT).

打开一个项目会执行下列步骤:

- 如果当前有打开项目,先关闭它
- 设开发模式

- 设处理器状态
- 打开并保存所有窗口到它们的保存状态
- 下载 hex 文件到仿真器存储器
- 复位处理器

要关闭一个项目,选择 *Project>Close Project*. 你会被询问在关闭它之前是否保存.

随着一个项目保存的信息有:

- 开发模式和处理器
- 时钟源和频率
- 与项目相关的源文件
- 最后 PIC 可执行文件的名称
- 打开的窗口以及它们的尺寸和位置
- 命名的断点设置
- 配置位设置

第五章　基本特性

5.1　介绍

MPLAB ICE 提供了丰富的工具用于应用的仿真和调试. MPLAB ICE 提供了一套基本的在线调试工具,可以用于运行,停止,复位和单步执行.更多的开发工具可以支持更高级的调试技能.

MPLAB ICE 仿真器的基本特性包括在 MPLAB IDE 软件中.这里提供了一个通用的描述,但更多细节请参考 MPLAB 用户手册(DS51025).

5.2　要点

MPLAB ICE 基本特性包括:

- 复位处理器
- 查看处理器存储器
- 查看文件
- 使用状态栏和工具栏
- 开始和停止仿真
- 使用软件断点
- 使用硬件断点
- 使用触发输入/输出设置

5.3　复位处理器

有三种方式复位处理器:



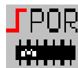
Debug>Run>Reset or **F9**

执行 MCLR,将程序计数器设置到复位向量



Debug>System Reset or **Ctrl-Shift-F3**

复位整个仿真系统,包括硬件,软件和目标处理器.

Power-On Reset 

Debug>Power-On Reset or **Ctrl-Shift-F5**

执行上电复位,类似于给一个器件上电,随机初始化 RAM.

5.4 查看处理器存储器

MPLAB 提供窗口用于查看各种存储器信息

程序存储器

程序存储器可以通过选择 *Window>Program Memory* 来查看.

EEPROM 存储器

如果被仿真器件包含 EEPROM,那么 EEPROM 的内容可以通过选择 *Window>EEPROM Memory* 来查看.

校验存储器

如果被仿真器件包含校验存储器,那么校验存储器可以通过选择 *Window>Calibration Memory* 来查看.这个窗口根据不同的仿真器件看起来会不一样.

文件寄存器

文件寄存器可以通过选择 *Window>File Registers* 来查看.这个窗口显示被仿真器件所有的文件寄存器.

特殊功能寄存器(SFRs)

特殊功能寄存器可以通过选择 *Window>Special Function Registers* 来显示.这个窗口提供的格式对于查看 **SFRs** 会比普通文件寄存器窗口更方便,因为每个 **SFR** 的名称都包含在内,并且提供了好几种数字的格式.

堆栈

堆栈的内容可以通过选择 *Window>Stack* 来查看.

> 注意: 如果堆栈中断使能被设置,MPLAB 在堆栈溢出和即将溢出时会显示警告.这个功能并不支持所有处理器模块.

观察窗口

要修改存储器内容,选择 *Window>Modify*,使用 Modify 对话框进行修改.

5.5 查看文件

当使用一个项目来调试代码时,当前执行点会由源文件和绝对列表文件来跟踪.要打开源文件,选择 *File>Open*,再选择相应的文件.在项目建立之后,绝对列表文件可以通过 *Window>Absolute Listing* 打开.表示当前执行的行会在每个窗口加亮.

5.6 使用状态栏和工具栏

MPLAB 状态信息在状态栏中显示出,而状态栏在 MPLAB IDE 窗口的底端.

处于便利,MPLAB IDE 包含四种工具栏,提供了执行程序任务的快捷方式.这四种工具栏是:

- 编辑工具栏
- 调试工具栏
- 项目工具栏
- 用户定义工具栏

点击工具栏最左边的按钮显示期望的工具栏.每个工具栏上的按钮可以根据你的需要重新配置.

5.7 开始和停止仿真

Run (green light)

Debug>Run>Run

从当前程序计数器位置开始仿真

Halt (red light)

Debug>Run>Halt

中止程序执行,并更新所有显示仿真信息.

Step

Debug>Run>Step

在当前程序计数器位置执行指令,更新所有显示仿真信息.

> 注意: 不要单步执行到 SLEEP 指令.如果你做了,你需要选择 **Debug>System Reset** 来唤醒处理器模块.

Step Over

Debug>Run>Step Over

Step Over 在功能上与 Step 一样,除了当执行指令是 CALL.在这种情况, Step Over 实时执行被调用的子程序,并在 call 的下一个地址中止.

执行一个操作数

Debug>Execute>Execute an Opcode

不修改程序存储器,执行一条或多条指令.

5.8 使用软件断点

MPLAB ICE 使用软件断点来使处理器在一特定位置中止.通过软件断点,在被指定中断的指令执行前,执行停止.

5.8.1 鼠标右键软件断点

将鼠标指在期望设置断点的指令上,通过点击鼠标右键来设置软件断点.点击右键出现菜单如图.

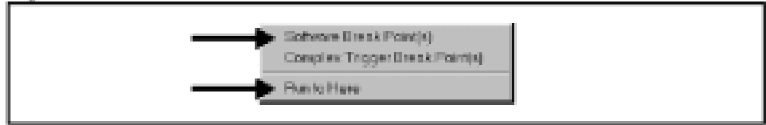


Figure 5.1: Right Mouse Button Menu - Software Break Point(s)

选择软件断点来设置软件断点.选择运行到这设一个临时软件断点.

断点可以在源程序,绝对列表或程序存储器窗口中设置.断点会在所有这些窗口中显示.

> 注意 1: 当项目重建或关闭时,以这种方式设的断点会丢掉.
> 2: 典型的,如果对外部存储器表写可以的话,软件断点可以在外部存储器代码执行地址设置.
> 3: 如果你不能设置软件断点,你可以使用复杂触发设置硬件断点.

5.8.2 命名软件断点

MPLAB 运行指定最多 16 个命名软件断点.这些断点可以选择性的使能和关闭.以这种方式设置的断点保存在项目中.

选择 *Debug>Break Settings* 显示断点设置对话框.

5.9 使用硬件断点

除了在程序存储器地址设置软件断点外,也可以通过更复杂的条件设置硬件断点.并且,硬件断点可以用于捕捉实时事件.

使用硬件断点,处理器在断点中止时,可能已经执行过断点指令一条或更多条指令.

5.9.1 鼠标右键硬件断点

单个复杂触发断点可以通过使用鼠标右键来设置.在一个源程序窗口,一个列表窗口,或程序存储器窗口,通过点击和拖拽选择期望的行.然后在鼠标右键菜单中选择复杂触发断点.



Figure 5.2: Right Mouse Button Menu - Complex Trigger

这样就会使用 **Ignore FNOP Cycles** 和 **Halt on Trigger** 设置和应用一个单事件顺序触发.

地址可以通过重复这个过程来从说明中移走. 如果一个复杂触发已被指定,并且它和断点说明发生冲突,你会被询问验证这个操作.

> 当项目重建或关闭时,以这种方式设的断点会丢掉.

5.9.2 复杂触发断点

复杂触发可以设置为在一个中断发生前要求最多四个事件发生.这些事件可以通过这三种方式组合:

- 顺序
- 所有事件
- 任何事件

另外,复杂触发特性使跟踪存储器窗口可以用于执行下列功能:

- 事件之间的计时
- 过滤跟踪

复杂触发断点可以选择性的使能和关闭.以这种方式设置的断点可以保存在项目中.

选择 *Debug>Complex Trigger Settings* 显示复杂触发设置对话框.

更多信息请参考 6.3 节.

5.10 使用触发输入/输出设置

MPLAB ICE 提供了下列外部输入和输出选项:

- 可以在最终触发事件或过滤跟踪事件上,产生一个非特定持续时间的单脉冲.
- 在外部触发输入的指定端中断.
- 在外部触发输入的上升沿凝固跟踪缓冲.

这些选项可以通过触发输入/输出设置对话框(*Debug>Trigger In/Out Setting*)来设置.

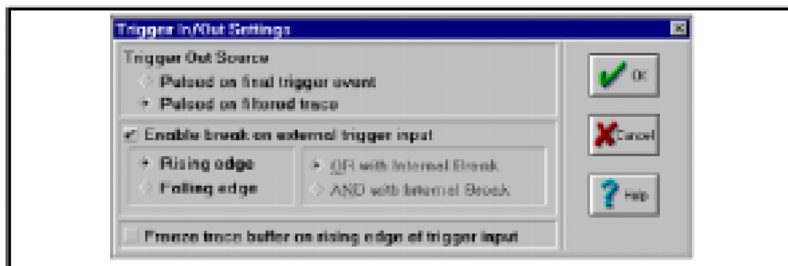


Figure 5.3: Trigger In/Out Settings Dialog

触发输入/输出设置可以用于逻辑探针.(B.6 节).

第六章 高级特性

6.1 简介

MPLAB ICE 提供了多种工具来仿真和调试一个应用. MPLAB ICE 提供了一套基本的在线调试工具,它具有运行,停止,复位和单步执行的功能. 另外更多的开发工具具有更多的高级调试特性.

当使用 MPLAB ICE 仿真器时,在 MPLAB IDE 中可以使用更多特殊的特性.关于这些特性的通用描述可以参考 MPLAB 用户手册,但是更多的细节在这里讨论.

6.2 要点

MPLAB ICE 高级特性包括:

- 使用复杂触发功能
- 复杂触发设置对话框
- 对话框语法
- 存储器访问选择
- 事件选择
- 复杂触发例子
- 使用鼠标右键进行复杂触发断点设置
- 使用代码覆盖
- 使用跟踪存储器窗口
- 查看跟踪存储器窗口
- 个性化跟踪存储器窗口

6.3 使用复杂触发

MPLAB ICE 具有高度灵活和功能强大的触发机制.一个触发是一些事件的组合,它可以导致:

- 一个硬件断点，与/或
- 一个跟踪存储器捕捉

一个事件是指在一个周期内捕捉到系统状态的一次描述. 另外,当触发发生时,可以产生一个外部信号. 这对于其它模拟器件或设备与 MPLAB ICE 同步是很有用的.

单个复杂触发断点可以使用鼠标右键来设置. 其它复杂触发功能需要通过复杂触发设置对话框来设置.

6.3.1　复杂触发设置对话框

复杂触发可以通过选择 Debug>Complex Trigger Settings 来指定. 根据你的选择,复杂触发设置对话框看起来会不一样:

1. 存储器访问 – 程序存储器或数据存储器
2. 事件 – 顺序事件,所有事件,任何事件,事件之间的计时或过滤跟踪

下图显示了当选择程序存储器访问和顺序事件时对话框的样子,这是对话框典型的样子.

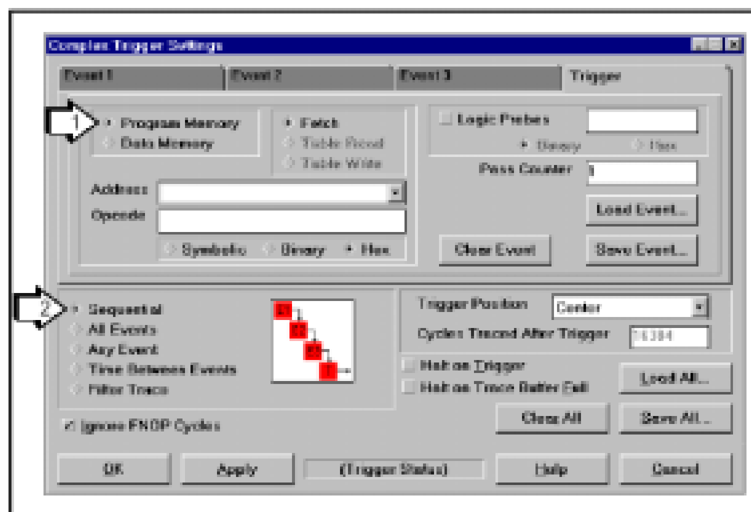


Figure 6.1: Complex Trigger - Program Memory, Sequential

关于存储器访问选择的信息,请看 6.3.3 节.更多存储器信息有:

- **地址(可选的)** – 一个单事件可以指定一个或多个地址.这可以是程序存储器或数据存储器地址.
- **操作数**或值(可选的) – 一个操作数的实际值,用于表读/写操作的数据, 或一个文件寄存器的值. 也选择这个操作数/值是作为符号,二进制或十六进制.

你可能进入的其它触发信息是:

- **逻辑探针(可选)** – 在外部逻辑探针输入的值. 也选择这个值是作为二进制或十六进制.
- **通过计数器** – 通过计数器计算在下一个事件进行前,这个事件必须发生的次数. 通过计数器只能在顺序触发类型中使用.

关于事件选择的更多信息,请看 6.3.4 节.

在接下来的对话框选项中,会遇到与跟踪相关的触发信息:

- **触发位置** – 用于定位在跟踪存储器窗口中的触发位置,显示触发发生后跟踪存储器窗口捕捉到的周期数. 产生触发的周期会被加亮.
- **触发中止** – 触发点会产生一个硬件断点,中止处理器,并会是跟踪存储器窗口的最后入口. 要不停止目标并捕捉跟踪,不要选择这个选项.
- **跟踪缓存满中止** – 在跟踪缓存覆盖旧的数据前,触发会让它停止. 当跟踪缓存满了时,硬件断点会中止处理器.

注意: 在配置跟踪对话框中指定的下载行数不影响采集的周期数,并且触发位置依然有效.

对话框更多选项:

• **Ignore FNOP Cycles** – 指定被强制的 NOP 周期是否在事件处理中被考虑. 一个强制 NOP 周期是指一条两周期指令的第二个周期.由于 PIC 单片机架构是流水线的,当它执行当前指令时,它会预取指物理地址空间的下一条指令. 然而,如果当前指令改变程序计数器,预取指指令会被忽略掉,这就导致了强制 NOP 周期.

假设一个项目有下列源代码:

```
RoutineA
                <code for RoutineA>
                RETLW    0
RoutineB
                <code for RoutineA>
                RETLW    0
```

当 RETLW 表示 RoutineA 被执行时,它的下一个地址空间的指令会被预取指(RoutineB 的第一条指令).

这条预取指指令不会被执行,但程序存储器地址出现在总线上. 如果在程序存储器地址 RoutineB 上设一个触发,那么在执行 RoutineA 中的 RETLW 期间的预取指会导致触发发生.要阻止它发生,可以检查 Ignore FNOP Cycles 检验栏. 当使用这个检验栏时需要考虑两点:

- 根据处理器模块,触发可能会在一个周期后
- 当对 12 位或 16 位内核的处理器模块使用过滤跟踪时, Ignore FNOP Cycles 不能被指定.

• **触发状态** – 当处理器运行时,这一区域显示当前定义的触发的状态.例子信息是:

- 没有触发使用 – 没有触发应用
- 事件 3:7 – 当前正在处理顺序类型触发
- 事件 3,在这个事件满足前,必须发生过七次以上.
- 在过程中 – 当前正在进行一个非顺序类型触发
- 完成 – 触发启动

在复杂触发设置对话框上有几个按钮,它们分别是:

• **载入事件** – 打开载入当前触发等级对话框,允许你载入一个*.trl 文件,它带有用于激活标签对话框的触发信息.
• **保存事件** – 打开保存当前触发级别对话框,允许你保存为一个*.trl 文件.
• **清除事件** – 清除当前触发信息.
• **全部载入** – 打开载入全部触发定义对话框,允许你载入一个*.trf 文件,它包含所有触发信息.
• **全部保存** – 打开保存所有触发定义对话框,允许你将触发信息保存为一个*.trf 文件.
• **清除全部** – 清除所有当前触发信息.
• **OK** – 接受当前设置并关闭对话框.
• **应用** – 接受当前设置,不关闭对话框.
• **取消** – 关闭对话框而不接受当前设置.
• **帮助** – 提供在线帮助文件,引导你设置一个复杂触发.

6.3.2 对话框语法

复杂触发地址语法		
描述	语法	例子
独立地址	<addr>	Delay
地址加/减偏移	<addr>+<offset> <addr>-<offset>	Delay+5 DelayEnd-2
地址范围	<addr1>:<addr2>	Delay:EndDelay Delay:Delay+5
两个或更多独立地址或范围	<addr1>;<addr2> <range1>;<range2>	Delay; EndDelay StartA:EndA;StartB:EndB
所有事情都在一个地址范围以外	!(<addr1>:<addr2>)	!(Delay:EndDelay) !(Main:Main+40)

每一个指定的地址可以是一个绝对地址,或者是定义符号.绝对地址必须以十六进制输入,并且首字符为一数字.

数值/操作数栏可以有两种输入方式.如果选择字符,那么根据上面的复杂触发地址语法所描述的,十六进制常数或标志符都能使用. 如果选择十六进制或二进制,那么只能有一个常数值可以输入. “**don’t care**”状态通过使用’x’或’X’指出. 如果基数是二进制, “**don’t care**”应用于一位.如果基数是十六进制, “**don’t care**”应用于四位.

外部输入值也是以二进制或十六进制输入.

通过计数器,捕捉事件数和触发位置值都以十进制输入.

复杂触发语法例子				
指定	**选择事件类型**	**放置**	**现场**	**数值类型**
程序存储器地址 0xAE26	程序存储器	0xAE26	地址	—
数据存储器地址 21	数据存储器	21	地址	—
所有 14 位 RETLW 指令	程序存储器	34xx	数值	十六进制
所有 14 位 RETLW 指令	程序存储器	3400:34FF	数值	字符
位 4 高,位 0 低	数据存储器	xxx1xxx0	数值	二进制

6.3.3 存储器访问选择

根据不同的存储器访问选择,复杂触发设置对话框的每一栏看起来会不一样.

6.3.3.1 程序存储器

如果指定了程序存储器访问,那么触发可以通过一个指令取指或一次表读/表写操作来产生.

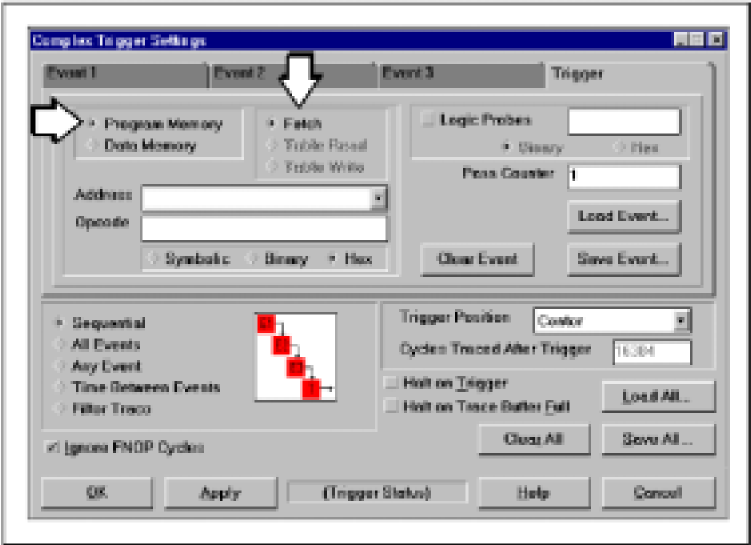


Figure 6.2: Complex Trigger - Program Memory Selection

6.3.3.2 数据存储器

如果指定了数据存储器访问,那么触发可以在一次读或写操作上设置给文件寄存器或特殊功能寄存器.这称作数据监测.

注意: 不支持数据监测的处理器模块(12 位内核的处理器模块)也不支持数据存储器触发.

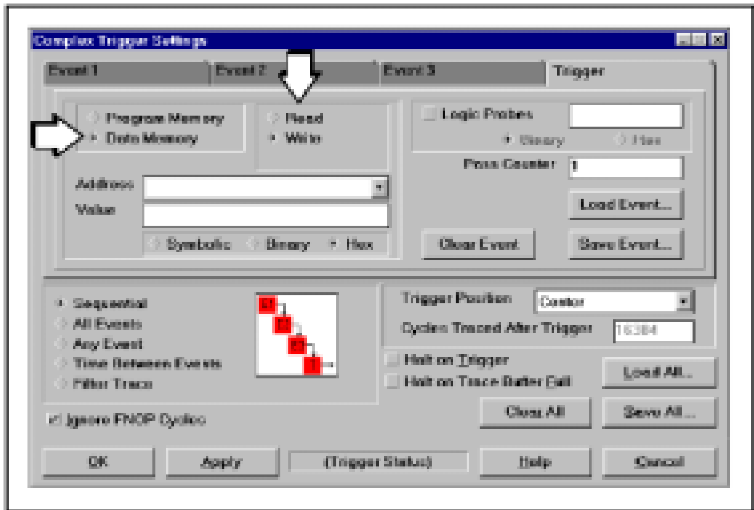


Figure 6.3: Complex Trigger - Data Memory Selection

6.3.4 事件选择

在跟踪或断点发生前,最多可以要求四个事件.这些事件的组合可以通过三种方式来指定:

- 顺序事件
- 所有事件
- 任何事件

另外,通过跟踪存储器窗口,复杂触发特性可以用于执行下列功能:

- 事件之间的计时
- 过滤跟踪

6.3.4.1 顺序事件

图 6.4 显示了复杂触发设置对话框的选择顺序事件的触发标签.

1. 如果每个事件必须按顺序发生来产生触发,选择顺序事件.
2. 点击一个标签,输入关于每个事件的信息(**事件 1, 事件 2, 事件 3**)和触发事件(触发).

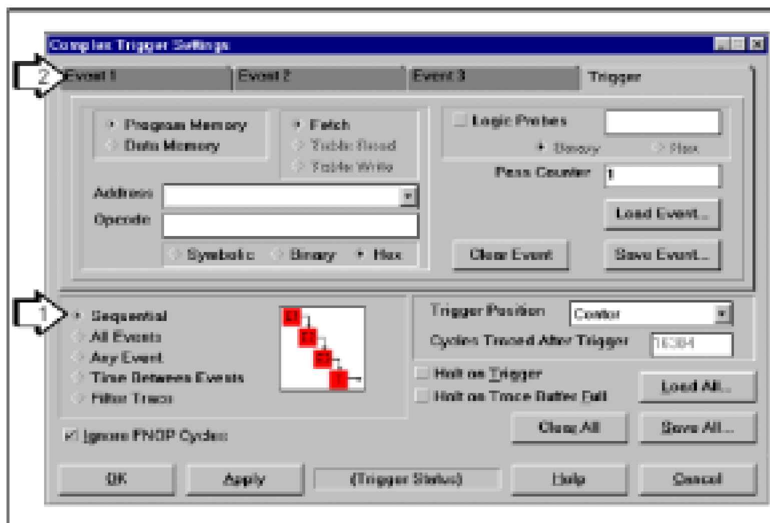


Figure 6.4: Complex Trigger - Sequential Event Selection

注意,对于顺序触发,在进行**事件 2** 之前,仿真器会等待直到**事件 1** 满足,依此类推直到最后的**触发事件**.如果只指定了一个触发事件,并把它放置在**事件 1** 上,那么在仿真器决定**事件 2,事件 3** 和**触发事件**满足时,额外的指令周期会被执行.因此,要利用右判断触发特性,也就是最先使用最右边的事件标签.

6.3.4.2 所有事件

图 6.5 显示了复杂触发设置对话框在选择所有事件时的事件标签.

1. 如果必须要每个事件发生来产生触发,并且顺序任意,那么选择所有事件.
2. 点击每个标签输入每个事件的信息(**事件 1,事件 2,事件 3,事件 4**).

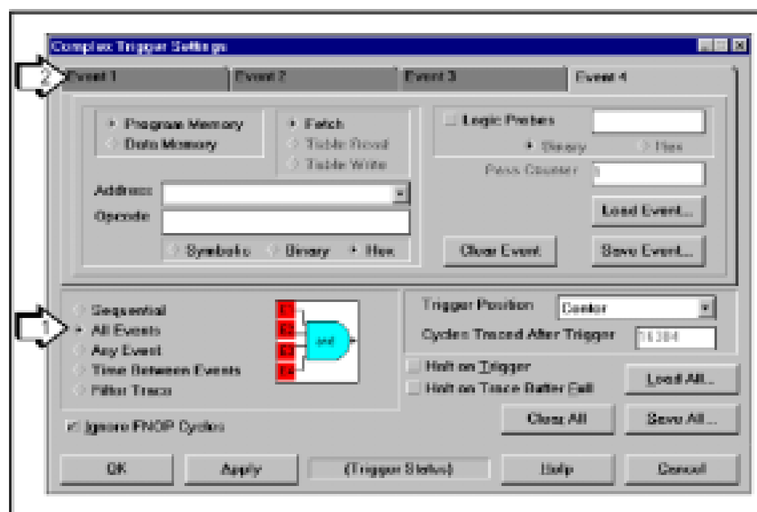


Figure 6.5: Complex Trigger - All Events Selection

对于这个事件选择,通过计数器不能应用(灰色的),因为对于触发事件计时不需要.

6.3.4.3 任何事件

图 6.6 显示了复杂触发设置对话框对于任意事件选择的一个事件标签.

1. 如果任何单个事件都会产生触发,选择任何事件.
2. 点击一个标签,输入每个事件的信息(**事件 1,事件 2,事件 3,事件 4**).

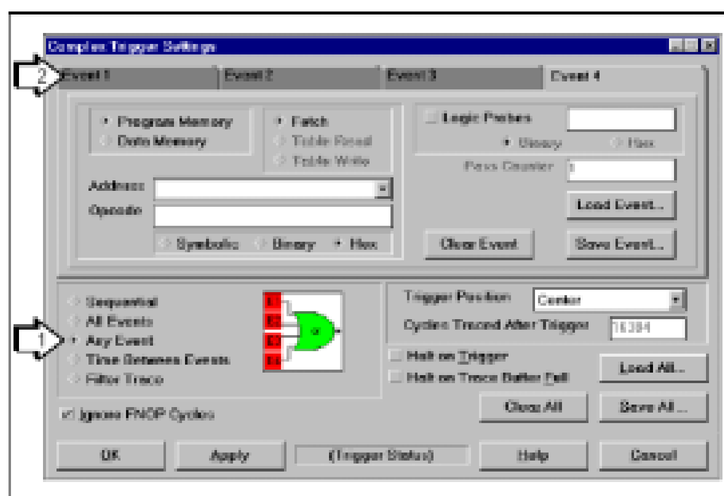


Figure 6.6: Complex Trigger - Any Event Selection

对于这个事件选择,通过计数器不能应用(灰色的),因为对于触发事件计时不需要.

6.3.4.4 事件之间的计时

图 6.7 显示了复杂触发设置对话框对于选择事件之间的计时的开始定时器标签.这个事件选择与跟踪存储器窗口相关联.

1. 选择事件之间计时，指定一个开始和中止事件.
2. 点击一个标签,输入相关信息:
   - 启动开始定时器的顺序事件(**事件 1，事件 2**)
   - 用于开始和停止定时器的事件(**开始定时器，停止定时器**)

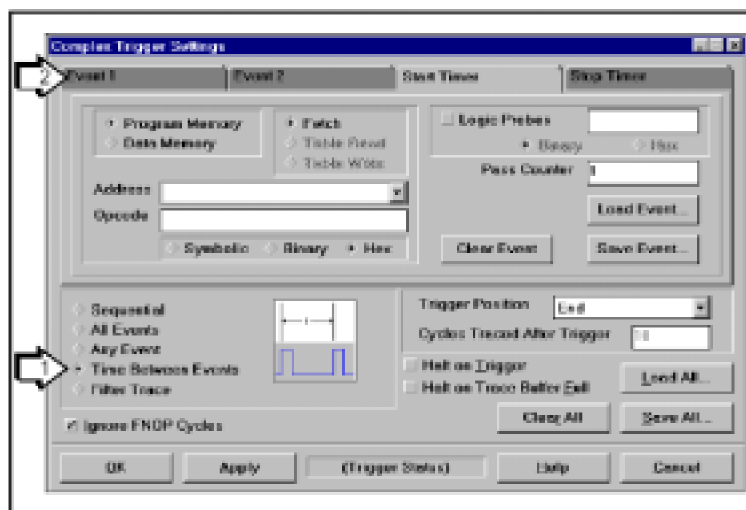


Figure 6.7: Complex Trigger - Time Between Events Selection

对于触发类型事件之间计时,计时标志发生器保持在零,直到一个指定的开始事件发生. 然后计时标志发生器连续递增直到一个指定的停止事件发生. 然后计时标志就可以用于测量逝去时间.

在计时标志发生器开始递增之前,最多两个事件和开始事件可以被指定顺序发生. 跟踪存储器显示了当开始事件发生时,计时标志开始递增,当中止事件发生时,计时标志停止.这样,通过检查跟踪存储器窗口的计时标志,就可以绝对事件之间的时间.

6.3.4.5 过滤跟踪

图 6.8 显示了复杂触发设置对话框对于事件之间计时选择的过滤标签. 这个事件选择与跟踪存储器窗口关联使用.

1. 选择过滤跟踪来指定事件,这些事件会通过跟踪存储器窗口来捕捉.
2. 要选择顺序事件来进行跟踪，点击一个标签输入关于事件的信息(**事件 1，事件 2，事件 3**).
3. 要执行连续跟踪,点击无穷事件.

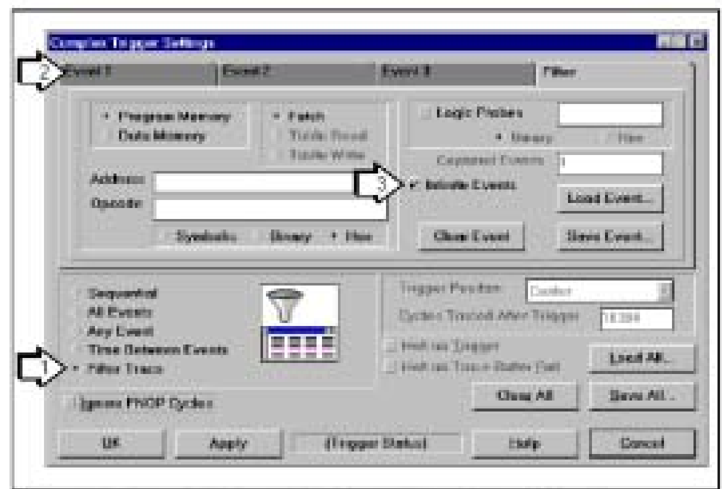


Figure 6.8: Complex Trigger - Filter Trace Selection

过滤器跟踪用于捕捉仅仅在跟踪缓存器中的某些事件.过滤跟踪允许你进行更多跟踪存储器的有效使用.你可以关注那些部分的代码,并仅仅采集那些,或者,你可以选择不收集延时循环或其它不感兴趣的重复代码部分.

在开始收集指定事件之前,最多有三个胜任的事件可以被指定顺序发生.在触发事件上的通过计数器栏变为被捕捉事件,显示这些事件记录的数量.

要执行连续过滤跟踪,检查无限事件检查框.无限事件时捕捉过滤跟踪的一个特殊情况.当跟踪缓存满了时,旧的数据会被放弃, 并按照数据先进先出的原则. 这在一些特殊的场合是很有用的.

1. 一直捕捉事件,直到用户中止.你可能希望看到数据被一直采集,直到系统被人为的中止.
2. 当你过滤事件来得到当前跟踪缓存的跟踪显示时,可以使用"重载数据".继续选择"重载数据",上载另一个当前被过滤的捕捉. 如果你在实时监测一个系统,并只是想看在不中止系统的情况下系统的运行情况,这时,这个功能就很有用了.
3. 有时,一个系统在运行很长一段时间后会误动作. 无限事件会在你的程序中继续采集数据,直到应用中止. 通常,误动作会使程序进入没有事件会被采集到的地方.然后,跟踪缓存器会在这个故障之前记录下这个感兴趣的数据.
4. 过滤跟踪可以在 TRGOUT 连接器上输出一系列触发. 如果你想在每个特定的程序被执行时,触发外部的逻辑分析仪,那么可以通过过滤那个地址并在触发输入/输出对话框中设置相应的触发来实现. 在这里,无限事件允许在 TRGOUT 连接器上产生连续的触发.

在过滤时触发位置是不可用的.

当执行一个过滤跟踪时,不可指定忽略 FNOP 周期,否则,跟踪缓存器会捕捉在引起触发的周期之后被执行的周期.

注意: 由于忽略 FNOP 周期不应和过滤跟踪一起被指定,所以预取指会导致触发发生.

同样,在数据存储器访问上执行一个过滤跟踪也是不推荐的,因为在触发指定后跟踪数据会是一个周期.

注意: 当多个事件触发时,注意在数据存储器访问上的触发会与触发它们的指令偏差两个周期.

6.3.5 复杂触发例子

下面的例子显示了复杂触发用于帮助调试或区分问题的几种方式.

6.3.5.1 顺序事件例子 – 程序存储器

一个应用有几个子程序.一个子程序(程序 A)正确的开始执行,但过了一段时间,它开始工作不正常.这个子程序被调用过许多次,因此可以跳过这个子程序工作正常的部分,而在这个子程序开始发生错误的地方中断.这里观察到这个程序工作正常,直到它之后的另一个子程序(程序 B)被调用.

这个问题可以通过两个事件的顺序触发来解决.必须发生的第一个事件是程序 B 的执行,因此,在**事件 3** 标签上指定程序存储器地址程序 B 的取指. 当程序 A 被调用时触发产生,因此在**触发**标签上指定对程序 A 程序存储器地址的取指. 忽略 FNOP 周期是左检查的,因此预取指被忽略掉,触发中止被检查,因此不正确的执行子程序会被单步执行.

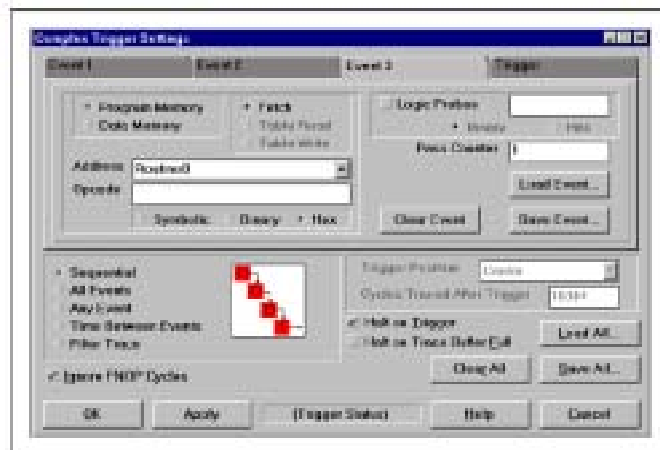


Figure 6.9: Setting the First of Two Sequential Events

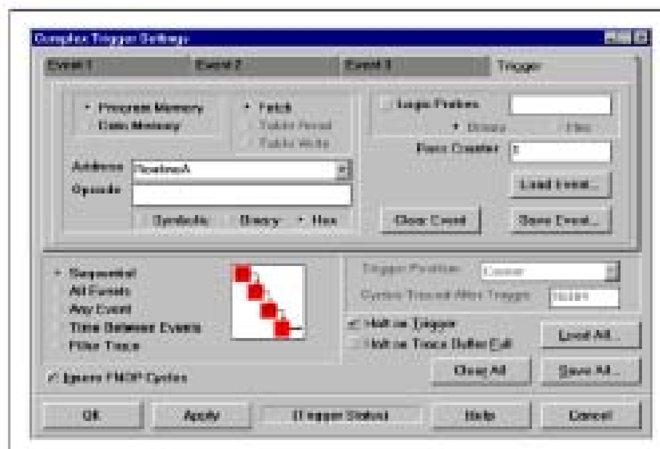


Figure 6.10: Setting the Second of Two Sequential Events

6.3.5.2 顺序事件例子 – 数据存储器

一个标志位在某个地方被错误的设置. 那么它是在那里被设置的?

触发也可以在数据存储器上被设置.假设这个标志位是在 RAM 区 Flags 的第三位.这里不关心其它位的值是什么,但是当 Flags 的第三位变成 1 时,触发应该产生. 对此,当修改数据存储器时使用一个顺序触发.指定地址为 Flags,数值为二进制 xxxx1xxx.. 如果愿意,可以不检查忽略 FNOP 周期,检查触发中止.

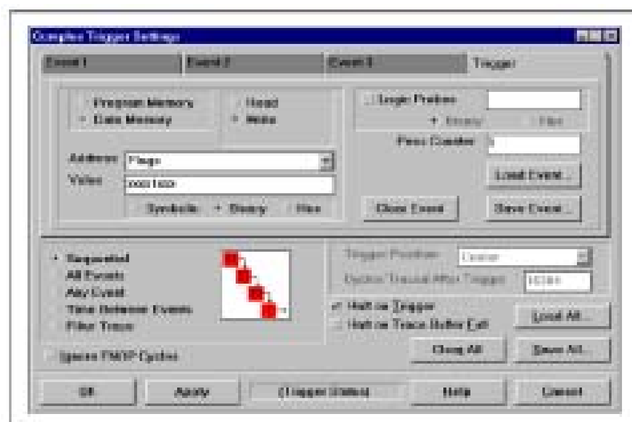


Figure 6.11: Sequential Event - Triggering on Data Memory

6.3.5.3 事件之间的计时例子

一个应用包含一个延时循环. 这个延时有多久?

对于这个问题,一个事件之间的计时触发是很有用的.假设这个延时程序叫做 Delay. 用于调试目的,在这个延时程序的返回声明地方添加一个标签 EndDelay. 设置**开始定时器**标签为对程序存储器地址 Delay 的取指,**停止定时器**标签为对程序存储器地址 EndDelay 的取指. 检查忽略 FNOP 周期框,让两个中止检查框不检查. 应用这个触发,打开跟踪存储器窗口,然后运行. 当运行到 EndDelay,跟踪存储器窗口会填满,最大的计时标志值就是两个事件之间的时间.

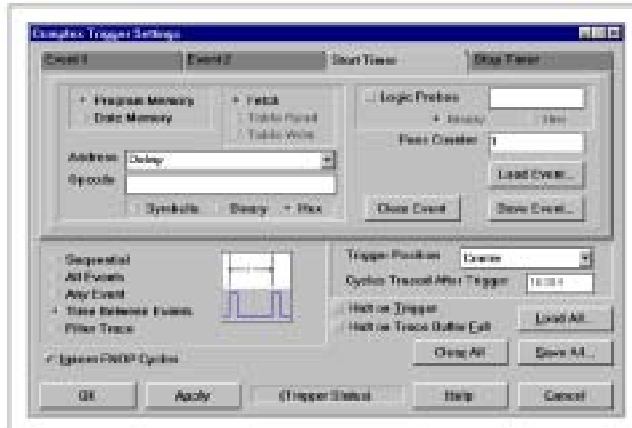


Figure 6.12: Specifying the Event that will Start the Timer

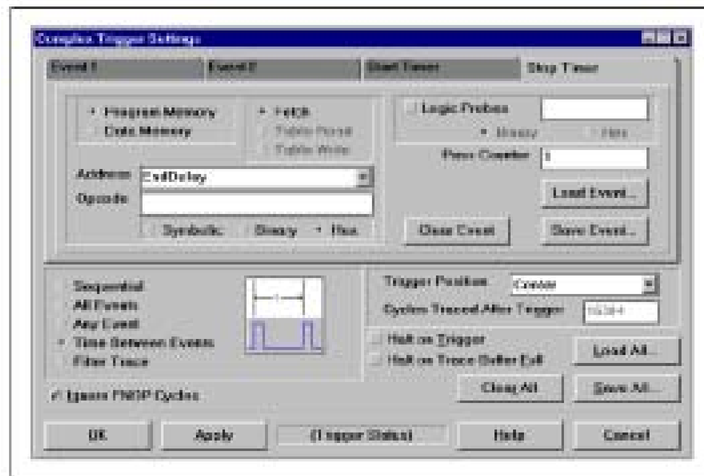


Figure 6.13: Specifying the Event that will Stop the Timer

6.3.5.4 过滤跟踪例子 – 程序存储器

一个程序有一个大的延时循环. 如何来跟踪程序,而不需要执行成千上万的延时循环指令?

对于这个问题,可以使用过滤跟踪.指定要跟踪的地址,除去延时循环所在的程序存储器范围. 不检查忽略 FNOP 周期. 检查无限事件,并使用一个软件断点或一个用户中断来中止仿真器或不检查无限事件,输入一个值用于被捕捉事件.当查看跟踪存储器窗口时,延时代码应已全部跳过去.

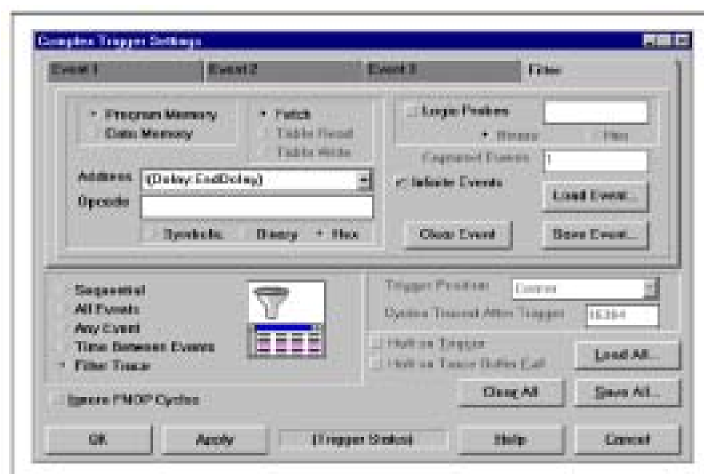


Figure 6.14: Filter Trace

6.3.5.5 过滤跟踪例子 – 数据存储器

如何对数据存储器进行过滤?

在 PIC 单片机文件寄存器读和写上的过滤会导致在 R/W 之后立即对周期进行采集. 但是这并不就是你要找的信息,

它提供了两个数据点:

1. R/W 发生之后的程序存储器，因此你可以跟踪到在你的代码中影响一个特殊变量或 SFR 的地方.
2. R/W 发生的次数，提供给你采集周期的数量.

使用一个过滤器可以有两种方法来捕捉实际的数据值. 第一种是在导致 R/W 操作的程序上进行过滤. 这会产生一些额外的周期,但会采集到正确的信息. 第二种方法是编译一段代码，这段代码在同一位置产生两个R/W周期,其中一个不影响实际的数. 例如,这个寄存器可以与 0xFF 这个值相与,来产生额外的周期:

```
test
    movlw     0xFF
    movwf     LSDigit
test1
    decfsz    LSDigit
    andwf     LSDigit    ; generate extra read/w to LSDigit
    goto      test1
```

6.4 使用代码覆盖

代码覆盖特性使代码被访问的部分(取指,写或读)可视化.这个被访问的代码在程序存储器窗口中被加亮,加亮的颜色和跟踪点文本一样,在 *Options>Environment Settings*，颜色标签上定义.

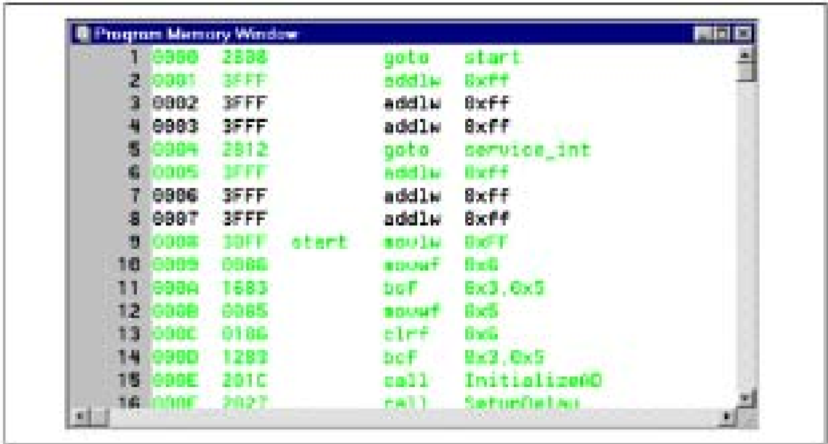
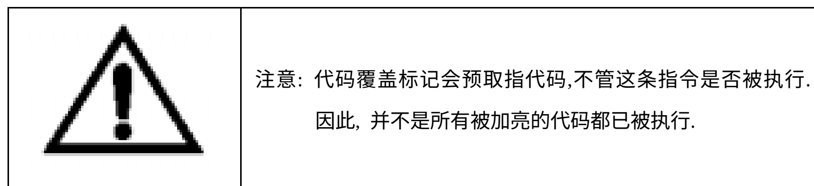


Figure 6.15: Program Memory Window - Code Coverage Enabled

当地址出现在总线中时,这个特性会通过锁存地址来工作. 因此,紧跟着双周期指令的修改程序计数器的指令可能并没有实际的执行.

在代码覆盖使能的情况下，仿真器遇到的下一个中止(软件断点或中止指令)会使在 ROM 区被取指的指令在程序存储器窗口中被加亮.



注意: 代码覆盖标记会预取指代码,不管这条指令是否被执行.
因此, 并不是所有被加亮的代码都已被执行.

使用这个方法, 作为一个表读(TBLRD)或表写(TBLWR)的结果被传送的地址会被跟踪.

要使能代码覆盖, 选择 *Debug>Code Coverage*.

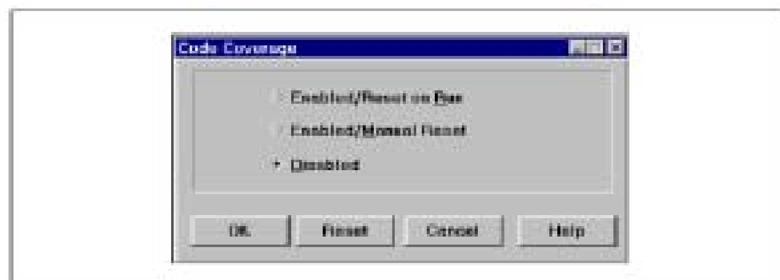


Figure 6.16: Code Coverage Dialog

如果运行**使能/复位**被选择,代码覆盖在每次仿真开始时被复位.如果**使能/手动**复位被选择,那么只有当一个复位被执行时,代码复位才被执行.

当产生一个复位,并且仿真已经运行,然后中止,所有被访问的程序存储器单元在程序存储器窗口中显示加亮.它们会保持高亮直到下次复位.单步执行不影响代码覆盖.

要关闭代码覆盖,选择 *Debug>Code Coverage*,点击 **Disabled**.

注意: 代码覆盖和复杂触发是相互独立的.当代码覆盖使能时,复杂触发就被禁止.当代码覆盖被禁止,以前定义过的触发必须重新应用.

6.5 使用跟踪存储器窗口

MPLAB ICE 中的跟踪存储器窗口包含的信息来自 MPLAB ICE 的跟踪缓存器. 跟踪缓存器由存储器硬件组成, 当 PIC 单片机执行指令时,它能记录各种 PIC 单片机数据,地址和控制信号的电气状态. 默认所有指令周期都通过跟踪缓存器捕捉. 复杂触发对话框可以用于控制哪些指令周期被捕捉.

在 MPLAB ICE 中,跟踪缓存器在一个 128 位宽的分析仪上捕捉数据,这个分析仪与仿真器的仿真芯片和其它器件相连.MPLAB ICE 分析仪捕捉的数据为:

- 程序存储器地址
- 程序存储器操作数/数据
- 文件寄存器源地址
- 文件寄存器源数据
- 文件寄存器目标地址
- 文件寄存器目标数据

- 外部逻辑探针
- 计时标志
- 额外内部仿真器信号

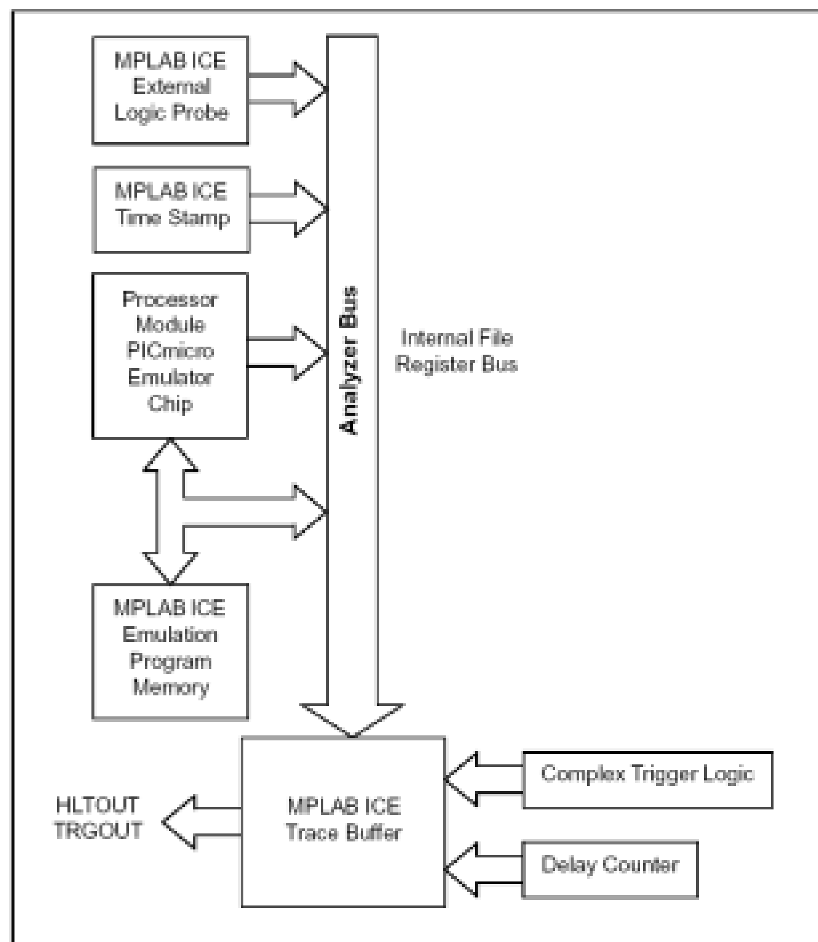


Figure 6.17: Trace Buffer Input – MPLAB ICE

6.5.1 查看跟踪存储器窗口

跟踪缓存器可以通过选择 *Window>Trace Memory* 来查看.

Trace Memory Window
File Data Options Help

	Addr	Op	Label	Instruction	SA	SD	DA	DD	Ex	Cycles
-16	0011	2811	loop	goto loop	—	—	—	—	11111111	0000714B57
-15	0012	1F0C		(Forced NOP	—	—	—	—	11111111	0000714B58
-14	0011	2811	loop	goto loop	—	—	—	—	11111111	0000714B59
-13	0012	1F0C		(Forced NOP	—	—	—	—	11111111	0000714B5A
-12	0011	2811	loop	goto loop	—	—	—	—	11111111	0000714B5B
-11	0012	1F0C		(Forced NOP	—	—	—	—	11111111	0000714B5C
-10	0011	2811	loop	goto loop	—	—	—	—	11111111	0000714B5D
-9	0012	1F0C		(Forced NOP	—	—	—	—	11111111	0000714B5E
-8	0011	2811		(Forced NOP	—	—	—	—	11111111	0000714B5F
-7	0004	2812		goto servic	—	—	—	—	11111111	0000714B60
-6	0005	3FFF		(Forced NOP	—	—	—	—	11111111	0000714B61
-5	0012	1F0C	service_ir	btfss 0xC,0x	—	—	—	—	11111111	0000714B62
-4	0013	0009		(Forced NOP	00C	40	—	—	11111111	0000714B63
-3	0014	081E		movf 0x1E,\	—	—	—	—	11111111	0000714B64
-2	0015	0086		movwf 0x6	01E	72	W	72	11111111	0000714B65
-1	0016	130C		bcf 0xC,0x6	006	72	006	72	11111111	0000714B66
0	0017	170B		bsf 0xB,0x6	00C	40	00C	–	11111111	0000714B67

Figure 6.18: Trace Memory Window – MPLAB ICE

> 注意: 由于处理器信号的计时,数据信息会偏离一个周期.目标数据值实际偏离两个周期,但是为了显示目的,跟踪缓存显示会补偿一个周期.

MPLAB ICE 提供一个跟踪存储器窗口监测很多的处理器操作.最多可以显示 32767 条指令.跟踪存储器窗口包含对于每个执行周期包含下列信息:

- 周期数 – 与触发或中止点相关的周期位置.
- 地址**(Addr)** – 从程序存储器被取指的指令的地址.
- 操作数**(Op)** – 被取指的指令.
- 标签**(Label)** – 与程序存储器地址相应的标签.
- 指令**(Instruction)** – 未编译的指令.
- 源数据地址**(SA)** – 源数据的地址或标志符.
- 源数据值**(SD)** – 源数据的值.
- 目标数据地址**(DA)** – 目标数据的地址或标志符.
- 目标数据值**(DD)** – 目标数据的值.
- 外部输入**(Ex)** – 外部输入的值.
- 计时标志**(Cycles或Seconds)** – 计时标志值.

近似的触发周期会用蓝色加亮,并标为周期 0. 所有其它的周期都以这个周期为基础来计数.在触发点之前发生的周期会记做负数, 在触发点之后发生的周期会记做正数. 如果在一次用户中止,一个软件断点,或一次复杂触发中止处理器之后,跟踪缓存器被显示,那么触发点会作为最后一个被跟踪的周期.当一个复杂触发填满跟踪缓存器而没有中止处理器,那么当复发触发发生时,触发点会显示近似的周期.

> **注意:** 当使用基于 12 位 PIC 内核的处理器模块时,原始和目标数据地址和值是不可用的.

在窗口中的菜单选项执行下列的功能:

• *File>Save* – 将显示的跟踪缓存信息保存到一个文件中.

• *Data>Find Trigger* – 将跟踪点转移到显示周期的中间.

• *Data>Reload* – 从仿真器重载跟踪信息.

• *Options>Configure* – 打开配置跟踪对话框,在6.5.2节中描述.

• *Options>Reset Time Stamp* – 复位计时标志的值.

• *Help* – 导出跟踪存储器的在线帮助.

选择 *Data>Reload*,强迫一次上载,而无论跟踪缓存器当前采集了什么数据.这允许你查看你的程序执行到了哪里.如果你的触发不能达到的话,这常常是很有用的.

6.5.2 自定义跟踪存储器窗口

纵宽可以通过使用鼠标拖拽右列边框来人工调整.选择 *Options>Configure* 可以进行更多跟踪显示的自定义.



Figure 6.19: Configure Trace Dialog

可配置选项包括:

• 每个纵列可以通过一个检查框来关闭.

• 未编译的指令可以以十六进制或字符格式显示.

• 数据地址可以以十六进制或字符格式显示.

• 外部输入可以以十六进制或二进制格式显示.

• 计时标志可以以周期数或秒数显示.

• 下载和显示的跟踪周期数可以选择.

配置跟踪显示的好处有:

• 关闭一个纵列可以阻止信息被上载到仿真器,这样可以加快显示时间.

• 使上载跟踪行的数目保持在最小,这也可以加快显示的速度.

要显示当前在仿真器跟踪存储器中的更多信息,可以增加上载跟踪行的数字,点击 **OK**,然后选择 *Data/Reload*.

6.5.3 读跟踪存储器窗口

读跟踪存储器窗口中的信息需要 PIC 单片机架构的知识.PIC 单片机指令取指一个指令周期,同时解码和执行前一个周期.

例一 (图 6.20):

1. 第一个周期,W 寄存器载入值 0xFF.
2. 下一个周期,W 寄存器的值写入变量.

Trace Memory Window

File Data Options Help

	Addr	Op	Label	Instruction	SA	SD	DA	DD	Ex	Cycles
-8	0000	2805		goto START	—	—	—	—	11111111	0000000000
-7	0001	3FFF		[Forced NOP]	—	—	—	—	11111111	0000000001
-6	0005	0186	START	clrf 0x6	—	—	—	—	11111111	0000000002
-5	0006	1683		bsf 0x3,0x5	006	00	086	00	11111111	0000000003
-4	0007	0186		clrf 0x6	003	1E	083	3E	11111111	0000000004
-3	0008	1283		bcf 0x3,0x5	086	FF	086	00	11111111	0000000005
-2	0009	30FF	LO	movlw 0xFF	083	3E	083	1E	11111111	0000000006
-1	000A	0086		movwf 0x6	—		W	FF	11111111	0000000007
0	000B	200F		call DELAY	006	0?	086	—	11111111	0000000008

Figure 6.20: Trace Memory Window – Example 1

例二 (图 6.21):

1. 第一个周期, 如果零应用到位置 0x20 的变量,递减文件并间跳.
2. 下一个周期, 这个变量的值从 2 递减到 1.

Trace Memory Window

File Data Options Help

	Addr	Op	Label	Instruction	SA	SD	DA	DD	Ex	Cycles
-9	0015	2814		goto DLOOP	020	03	020	02	11011111	000002FF00
-8	0016	0BA1		[Forced NOP]	—	—	—	—	11011111	000002FF01
-7	0014	0BA0	DL	decfsz 0x20	—	—	—	—	11011111	000002FF02
-6	0015	2814		goto DL	020	02	020	01	11011111	000002FF03
-5	0016	0BA1		[Forced NOP]	—	—	—	—	11011111	000002FF04
-4	0014	0BA0	DLOOP	decfsz 0x20	—	—	—	—	11011111	000002FF05
-3	0015	2814		[Forced NOP]	020	01	020	00	11011111	000002FF06
-2	0016	0BA1		decfsz 0x21	—	—	—	—	11011111	000002FF07
-1	0017	2814		[Forced NOP]	021	01	021	00	11011111	000002FF08
0	0018	0BA2		decfsz 0x22	—	—	—	—	11011111	000002FF09

Figure 6.21: Trace Memory Window – Example 2

第七章　校　　验

7.1　简介

这章描述了如何检验 MPLAB ICE 系统硬件工作正常.

7.2　要点

这章讨论了:

- 运行校验
- 常见校验失败的问题

7.3　运行校验

MPLAB ICE 校验是一个诊断工具,它是用于测试 MPLAB ICE 各个模块运行是否正常的.它运行时,系统除了插入一块处理器模块,应该与任何目标板断开.

7.3.1　开始验证

在 MPLAB IDE 安装目录下运行 verify.exe 来启动验证程序.验证程序的界面如图(图 7.1).

Figure 7.1: Verify Desktop

首先,你会被警告,如果有其它设备接在被验证的端口上,验证程序可能会损坏它们. 确保你没有通过一个开关盒 **(switchbox)**或过通**(pass-through)**来使用 MPLAB ICE,并且已经选择好用于 MPLAB ICE 正确的并行口.

> **注意:** 如果你为 MPLAB ICE 专门分配了一个端口,并不希望每次都看到这条信息,可以在检查框选择 " 关闭这个警告信息".

点击 **OK** 继续.点击 **Cancel**,以便在验证前检查你的连接,然后选择 *Tools>Verify MPLAB ICE*.

7.3.2 使用验证对话框

现在验证对话框会出现.在这个对话框中有三个标签:

- 访问测试 – 检查基本访问对象和系统的静态功能(图7.2).
- 运行模式测试 – 检查系统的运行(图7.3).
- 版本 – 显示MPLAB ICE配置信息(图7.4).

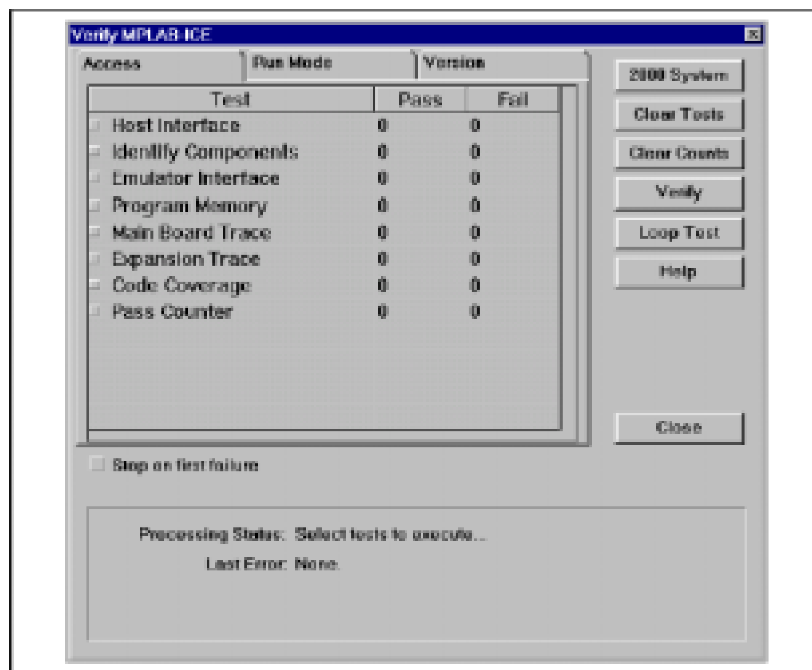


Figure 7.2: Verify MPLAB ICE – Access Tests

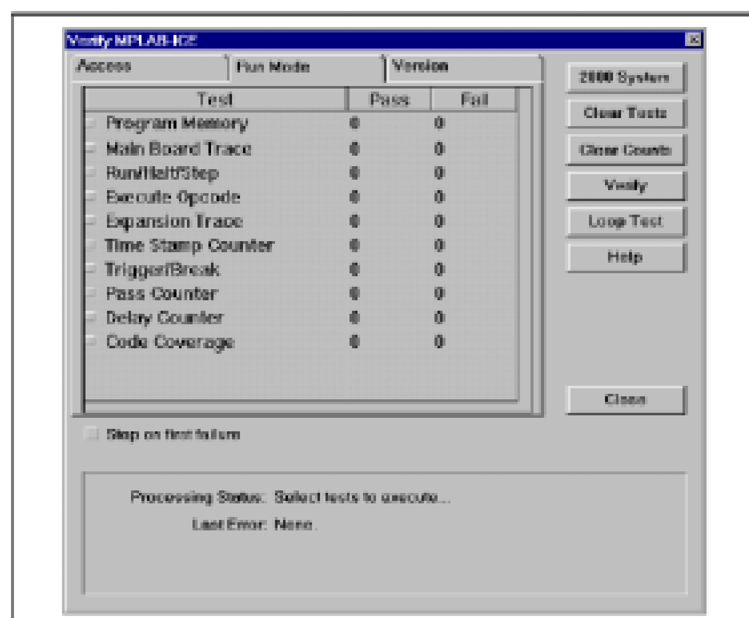


Figure 7.3: Verify MPLAB ICE – Run Mode Tests



Figure 7.4: Verify MPLAB ICE – Version Information

当对话框出现时,所有左侧的检查框默认都是未检测的.通过和失败计数器应该显示每次测试运行通过和失败的次数.这些按钮执行下列功能:

- **2000 System** – 检查所有属于MPLAB ICE 2000仿真器系统的测试.
- **Clear Tests** – 不监测所有测试检查框
- **Clear Counts** – 设置所有通过和失败计数值为零.
- **Verify** – 执行所有待检查的测试.
- **Loop** Test – 反复执行所有待检查的测试,直到Stop Loop被点击.
- **Help** – 打开MPLAB ICE校验的在线帮助.
- **Stop on first failure** – 如果检查到错误,测试会在第一个失败中止.
- **Close** – 关闭对话框.在一次测试的中间将不起作用.

要运行一组校验测试,可以分别选择期望的测试,或使用**2000 System**选择所有测试. 然后,选择**Verify**开始运行测试.当它运行时,MPLAB在每个测试过程会显示信息.当发现任何不正常的动作,它会报告出来. 要反复运行所有被选择的测试,选择**Loop Test**,而不是**Verify**.

7.3.3 使用端口选择对话框

在关闭校验对话框时,端口选择对话框会打开.这个对话框也可以从*Tools>Port/Mode select*来打开.

这个对话框有两个标签:

- Ports – 设置与MPLAB ICE进行通讯的LPT端口(图7.5).
- Tools – 设置工具用于校验(图7.6).

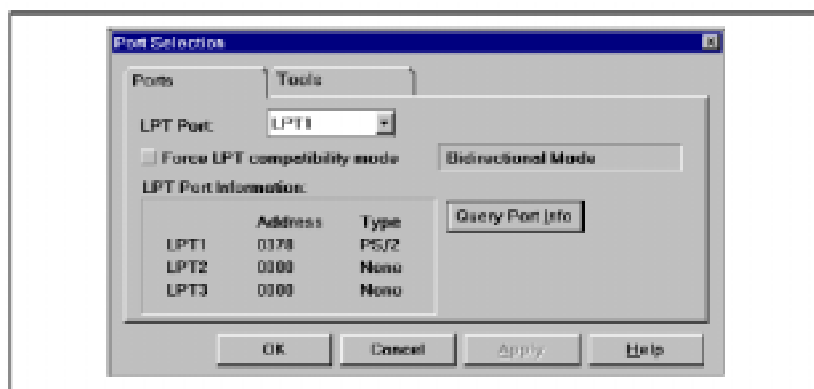


Figure 7.5: Port Selection Dialog - Ports Tab

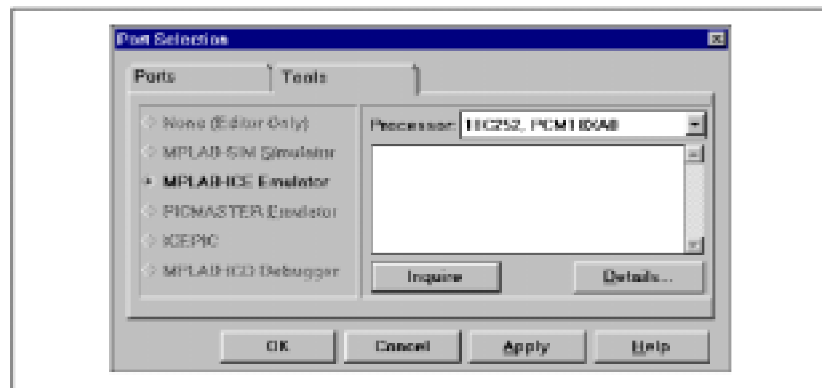


Figure 7.6: Port Selection Dialog - Tools Tab

如果你在验证MPLAB ICE时有问题,你可以使用**Ports**标签来设置一个不同的端口.如果你的MPLAB ICE有通讯问题,请参考第八章.

选择要验证的工具(只有MPLAB ICE可选),使用**Tools**标签.点击**Details**,查看器件限制的相关信息.

点击**OK**关闭并保存你的更改.点击**Cancel**关闭而不保持.点击**Help**查看在线帮助信息.

7.3.4 启动及退出验证程序

有时你关闭了验证对话框,你可能会使用*Tool* 菜单的选项重新打开:

- *Verify MPLAB ICE* – 打开验证MPLAB ICE对话框
- *Port/Mode select* – 打开端口选择对话框

当你完成了验证,可以通过下面方法的一种来退出程序:

- 选择*File>Exit*
- 使用组合键**Alt-F**或**Alt-F4**
- 从系统菜单按钮选择关闭(视窗左上角).
- 点击窗口右上角的"**X**"

7.4 校验失败常见问题

当运行校验时,如果你的MPLAB ICE报告失败,请检查下列选项:

- 确认你的MPLAB ICE单元的电源打开,并与你的PC正确的通讯.要判断这点,选择*Tools>MPLAB ICE Configuration*,然后从这个对话框的MPLAB ICE系统标签选择**Inquire**. 要解决通讯问题,请参考第八章.
- 确认处理器模块正确的插入MPLAB ICE单元.(错误灯不应被点亮).也要确认处理器模块没有和目标板连接.
- 确认处理器模块正确的配置为一个有效的处理器.如何配置你的处理器模块,请参考第四章.

如果你检查了上面几条,并仍然认为你的MPLAB ICE系统运行不正确,请按照通用信息的向导栏来获得客户支持,以便对你的工具进行维修或更换.

第八章　常见问题

8.1　介绍

这章描述了运行 MPLAB ICE 的一些常见问题以及如何解决这些问题.

8.2　要点

这章讨论了下列要点:

- 常见问题
  - 不能和 MPLAB ICE 建立通讯
  - 非法并口选择(Windows NT 4.0)
  - MPLAB ICE 驱动器未被载入(Windows NT 4.0/ Windows 2000)
  - 程序存储器看起来正确,但是文件寄存器和程序执行看起来工作不正常
  - 复位时,出现一个错误信息,说有一个错误,重置处理器并检查电源
  - 电源灯闪烁
- 为 MPLAB ICE 配置一个 PC 并口接口

8.3　常见问题

不能和 MPLAB ICE 建立通讯

检查 LPT 端口

- 验证 MPLAB ICE 连接到被选择的 LPT 端口.
- 选择 *Options>Development Mode*,并选择 **Port** 标签.点击 **Query Port Info** 查看可供使用的 LPT.
- 验证 LPT 端口和 MPLAB 软件使用的是相同的通讯协议.通过 PC 的 BIOS 检查 LPT 端口设置.如果使用一个外接卡(add-in card),验证卡的跳线.为了优化执行效率,将你的 LPT 端口设为双向模式(即 PS/2,EPP,或 ECP).如果 MPLAB 认为端口支持它,它会使用双向通讯模式.
- 如果你的并口不支持双向模式,可以尝试让 MPLAB 使用可兼容模式.选择并检查 **Force Compatibility Mode** 检查框.
- 在 Windows 2000,操作系统可能会为 MPLAB ICE 的 LPT 端口报告一个与 MPLAB 使用的不同的地址.如果出现这种情况,你必须更改操作系统下的 *Control Panel>System>Device Manager* 的设置.点击+查看端口,双击它,或选择它然后点击属性来设置 **NT** 里的地址.
- 如果仍然不起作用,继续 8.4 节的为 MPLAB ICE 配置 PC 的并行口接口.

检查目标时钟/电源

- 如果你正在使用目标时钟,选择仿真器时钟(Options>Development Mode>Clock,不检查 Use Target Board Clock),再看你是否能建立通讯.如果能,那么可能是目标时钟的问题.
- 如果你在使用目标电源,选择仿真器电源(*Options>Development Mode>Power*,选择 Processor Power From Emulator),再看是否能建立通讯.如果能,可能就是目标电源的问题.

非法并口选择(Windows NT 4.0)

在 Windows NT4.0 中,检查 PC 的 BIOS 的并行口设为 ECP 模式,并且可兼容模式没有选择.在一些 PC 的 BIOS 中,ECP 模式被标为 PS2 或普通模式.端口不应设为 SPP 或 EPP 模式.启动 PC 来查看你的 PC 的 BIOS.按照显示器提示的按键进入 SETUP 界面,并按提示进行操作.要查看 MPLAB ICE 驱动器是否运行,可以打开 *Control Panel>Devices*,并滚动屏幕查找 MPLAB ICE.

MPLAB ICE 驱动器未被载入(Windows NT 4.0/ Windows 2000)

如果你怀疑 MPLAB ICE 驱动器未被正确的载入,可以检查 Event Viewer. 在 Windows NT 4.0 中,Event Viewer 在 *Start>Programs>Administrative Tools* 下.在 Windows 2000 中, Event Viewer 在控制面板中.”mplabice”应该出现在 Event Viewer’s System Log 中.双击 mplabice 所在的行查看事件属性.它应该显示 MPLAB ICE 驱动器被成功载入.如果在 Event Viewer’s System Log 中没有”mplabice”,驱动器就是没有被载入.

程序存储器看起来正确,但是文件寄存器和程序执行看起来工作不正常

- 确信处理器时钟设为正确的速度.选择*Options>Processor Setup>Clock Frequency*,打开Processor Clock对话框.验证对于被仿真的器件,在当前操作电压下的频率是否在正确的范围内.
- 通过选择*Options>Processor Setup>Hardware*和验证处理器电源选项来检验是否选择正确的电源.如果使用目标板作为供电电源,检验目标板电源是否打开.

复位时,出现一个错误信息,说有一个错误,重置处理器并检查电源

如果你正在使用低电压仿真,在连接目标板并打开电源的情况下,进行一次系统复位后,这条信息可能会出现.然而,MPLAB和MPLAB ICE在第一次尝试时并不总是保持同步.在绝大多数情况下,点击**Yes**重试会工作正常,初始化会继续.如果在两三次后仍不工作,点击**No**,MPLAB会尝试继续,并报告其它任何遇到的错误.

电源灯闪烁

在有些主机中,当有系统故障时电源灯会闪烁.关闭主机然后打开以清除故障.如果这样没有清除故障,请联系Microchip支持.

8.4 为MPLAB ICE配置PC的并行口接口

MPLAB ICE仿真器使用工业标准,双向并行外围接口. 大多数PCs只配置了一个并行接口,但根据PC可使用的插槽可以配置到三个.

MPLAB ICE可以以可兼容模式和双向模式进行通讯.根据不同的电脑或并口生产厂商的功能实现,可能有一种或两种能够工作.

绝大多数情况,MPLAB ICE在不需要用户的干预下,可以自动识别并和安装的并行接口进行通讯.但你也可能会碰到一些特殊的电脑,要求你来配置并行接口,以便它能正确的和MPLAB ICE通讯.

8.4.1 检查PC BIOS设置

当配置一台电脑来运行MPLAB ICE, 检查并记录下你的电脑BIOS中的并口特性.

通常,在电脑重启时,你可以按一个键或一组键来进入BIOS配置设置.当电脑启动时,电脑的监视器会显示通过使用什么键来进入BIOS配置设置.例如,对于Hewlett Packard,当你第一次打开电脑时,它会显示”<F2>Setup”. 通常是<F1>,<F2>,<F10>,或<DEL>. 当你启动电脑时观察监视器,或者参考你的电脑厂商的文档来找到如何访问BIOS配置设置.

尽管并行接口是工业标准,但协议却是独立实现的.BIOS屏幕应该包含并口设置的信息.请记下BIOS默认的设置.

大多数BIOS设置屏幕都有它们使用的指令.由于操作系统还没有被载入,你必须使用键盘按键来进行选择,设置和保存你的变更.

通常,查看主设置,注意你的电脑是否允许即插即用的操作系统来配置设备.

然后,导航到”高级”(Hewlett Packard)或通信菜单,直到你找到端口配置设置.一个设置显示自动,是或否,它是用来显示LPT端口是通过即插即用操作系统来自动配置,通过BIOS来配置,或被禁止.如果你认为有些东西越出了你存在的设置,你可以选择是来确信你在BIOS中所做的设置都是有效果的.

模式选择通常在上面那个设置附近.通常会有”可兼容模式(compatible mode)”,”双向模式(bi-derectional mode)”,”EPP模式(EPP mode)”,或”ECP模式(ECP modes)”. 根据生产厂商,这些设置中的一种或多种可以和MPLAB ICE工作.有些特定的设置对于某些情况可能不能工作,所有有必要在以后返回这些设置.

查看如何改变设置的指令.通常你反复的按一个功能键来滚动可能的值,然后推出菜单.注意要知道哪个键是用来保存你的选择,并要避免按哪个功能键会将所有设置恢复成默认值.当你退出BIOS设置,你的电脑会继续启动.

如果你无法进入BIOS,或不熟悉BIOS设置,请参考你电脑的操作指南或找系统管理员.

8.4.2 检查操作系统设置

在检查你电脑的BIOS设置后,再检查与你电脑相匹配的操作系统的并口知识.在Windows 95中,从开始菜单选择设置,控制面板.双击系统图标.点击设备管理器标签,然后点击端口旁边的加号来展开端口设置.你应该看到在这儿列出的端口编号和BIOS中的相同.如果列出的没有在BIOS中的多,你可能需要给系统添加端口.双击控制面板的添加新硬件(参考你的操作系统文档).

如果有问题,请参考电脑操作系统文档或请教本地系统管理员.

8.4.3 配置MPLAB ICE

要鉴定并行端口是否正确的配置到你的MPLAB ICE,你可以使用MPLAB ICE校验程序(*Tools>Verify MPLAB ICE*).使用这个方法报告的错误可以显示失败的组件,但有些解释可以显示需要重新配置并口.

在了解了并行口的特性后,你会想确信MPLAB ICE与电脑连接并通讯上.安装完MPLAB ,选择*Options>Development Mode*和配置MPLAB ICE.

点击**开发**模式对话框的端口标签.MPLAB所识别的各种并口特性都列出在对话框中.点击**Query Port Info**.

这时会有一对话框提示你,在继续测试之前,从系统断开其它共享的并口设备.采取任何的预防措施都是必要的.然后点击**OK**.

任何系统识别的并行端口都会与它们的基地址和类型一起出现,就和MPLAB所识别的一样.在MPLAB和系统BIOS之间所识别的端口是相关联的,尽管在具体的术语方面会有一些不同.例如,BIOS报告一个端口是双向的,而MPLAB会进一步说明这个端口是PS/2.

注意: 我们建议你将LPT1用于MPLAB ICE.在某些情况下(特别是Windows NT和Windows 2000),MPLAB 无法识别更多的LPT端口,如LPT2.

如果你认为MPLAB ICE没有报告足够的端口(这种情况可能发生在你添加了一块并行接口卡之后),可能有必要给系统配置添加硬件设备.在Windows 95中,这要在系统对话框中操作.从开始菜单选择设置,控制面板.双击系统图标.点击设备管理器标签,然后点击端口旁边的加号展开端口设置.如果这儿列出的没有BIOS中的多,你可能需要给系统添加端口.在控制面板中双击添加新硬件(参考操作系统文档).

在MPLAB中,选择Options>Development Mode.选中工具标签,确认MPLAB ICE作为选中的工具,并且处理器模块无误.点击端口标签.点击查询端口信息,通过软件为MPLAB ICE系统查找已知的并行端口.如果它找到并能正确的通讯,MPLAB状态栏会更新显示工具(MPLAB ICE)和安装的处理器模块.

8.4.4 找不到MPLAB ICE

如果在你尝试配置MPLAB ICE,并点击开发模式对话框端口标签上的查询端口信息后,软件报告找不到MPLAB ICE,你应该进行如下操作.

1. 选择Options>Development Mode,点击端口标签.选择强制可兼容模式检查框,点击查询端口信息.
2. 如果MPLAB ICE现在被识别,那么你现在在可兼容模式.MPLAB ICE在这种模式应能工作正常,但是系统对电脑的响应会变慢.
3. 如果MPLAB ICE仍然不能找到,那么有可能是在并口如何配置,如何实现以及MPLAB ICE如何识别并口之间出现了冲突.

8.4.5 更正冲突

返回到BIOS设置界面.在设置界面上,定位在并行端口,特别是你认为与MPLAB ICE相匹配的那一个.

记下原始的信息.依照电脑厂商的指令,从提供的选项中更改端口属性.这里并没有绝对正确的答案,但是推荐你先选择的模式是仅仅作为输出,SPP,或可兼容模式.然后保存新设置.重新验证MPLAB ICE.

如果选择所有可能的设置后,你的MPLAB ICE仍然无法和电脑进行通讯,那么很可能并行端口是不标准的.尤其是消费类电脑端口是主板集成的,这种可能是很大的.这时有必要安装一块独立的并行接口卡专门用于MPLAB ICE.

如果你无法通过BIOS改变端口设置,那么它们可能是由一块独立的端口卡或卡上的跳线设置来控制.这些卡可以通过厂商的指令来进行配置.

在你改变并行接口特性之后,MPLAB ICE在验证并行口时会报告一个不同的值.

8.4.6 校验MPLAB ICE

你可能会发现不只一个并行端口设置允许你配置和查询MPLAB ICE端口信息.

根据并行端口的实现,可能会有一个或多个端口能更好的和仿真器系统工作.要决定是否做了最好的选择,运行Tools>Verify MPLAB ICE可能会很有用.如果报错,可以改为强制可兼容模式,或在系统BIOS对端口设置进行更改.

8.4.7 和其它产品共享并行接口

MPLAB ICE不应和其它设备共享并口.例如,将软件狗和MPLAB ICE一起使用会导致软件狗的损坏.将并行接口和其它需要驱动器载入的设备共享也会导致无法预料的结果.

如果你必须改变以前定义过的端口的特性,那么你首先要删掉这个设备和相应的驱动.然后重新配置端口并重新安装设备.

8.4.8 速度问题

以可兼容模式使用MPLAB ICE会导致和仿真器通讯的效率的下降.在上载和下载大量数据时尤为显著,但并不影响系统的实时性能.

一旦断定可兼容模式工作正常,那么就有必要尝试其它的并口模式(即,双向模式)来提高效率.

8.4.9 仍不工作

有可能并口的所有设置对MPLAB ICE都不起作用.即使是其它的并口设备(即打印机)在这个端口上能正常工作.这并不是MPLAB ICE的问题.MPLAB ICE是依靠标准IEEE来实现的,而打印机是依靠操作系统和并行端口进行接口来实现的.这种情况下,建议使用一块独立的并行接口卡.

附录 A　调试技巧

A.1　介绍

这部分附录描述了各种调试技巧,你在用 MPLAB ICE 仿真时可能会有用.

A.2　要点

这部分附录包含以下信息:

- 复杂触发例子
- 更多调试技巧

A.3　复杂触发例子

下面列出的调试技巧在以前的复杂触发例子中已经包含了.

A.3.1 子程序开始失败

一个应用有几个子程序.一个子程序(程序A)正确的开始执行,但过了一段时间,它开始工作不正常.这个子程序被调用过许多次,因此可以跳过这个子程序工作正常的部分,而在这个子程序开始发生错误的地方中断.这里观察到这个程序工作正常,直到它之后的另一个子程序(程序B)被调用.

请看顺序事件例子 – 程序存储器

A.3.2 标志位被错误置位

一个标志位在某个地方被错误的设置. 那么它是在那里被设置的?

请参考顺序事件例子 – 数据存储器.

A.3.3　延时循环的长度

一个应用包含一个延时循环. 这个延时有多久?

请参考事件之间的定时例子.

A.3.4　略过长延时的跟踪

一个程序有一个大的延时循环. 如何来跟踪程序,而不需要执行成千上万的延时循环指令?

请参考过滤跟踪例子 – 程序存储器.

A.3.5　过滤数据存储器

如何对数据存储器进行过滤?

参考过滤跟踪例子 – 数据存储器.

A.4 更多调试例子

A.4.1 中断之间的计时

一个应用包含一个中断.那么中断之间的时间有多长呢?

有两种方式来测试: 使用一个过滤的跟踪,或使用事件之间的计时.

方案A: 过滤跟踪

使用过滤跟踪,每次中断服务程序开始时进行捕捉.

要清除所有中断,跟踪和触发点,选择*Debug>Clear All Points*.选择*Debug>Complex Trigger Settings*打开复杂触发设置对话框.



Figure A.1: Filter Trace for ISR Time

设置复杂触发设置对话框用于程序存储器的过滤跟踪.忽略FNOP周期不应被选中.

输入ISR开始地址(可以是数字地址,也可以是标志符).在图A.1中,地址是irqsvc.

如果无限事件被选中,取消这个选项.然后输入捕捉事件的数目,或你想捕捉ISR开始到跟踪存储器窗口的次数.在图A.1中,捕捉事件是200.最后,选中跟踪缓存满中止.

点击**Apply**或**OK**.

复位系统(*Debug>System Reset*),然后运行程序(*Debug>Run>Run*,或工具栏上绿色灯图标).中止之后,打开跟踪存储器窗口(Window>Trace Memory).

Trace Memory Window

File Data Options Help

	Addr	Op	Label	Instruction	SA	SD	DA	DD	Ex	Seconds
0	0004	0000	irqsvc	nop	—	—	—	—	11011111	0.013332000
1	0004	0000	irqsvc	nop	—	—	—	—	11011111	0.013464000
2	0004	0000	irqsvc	nop	—	—	—	—	11011111	0.013596000
3	0004	0000	irqsvc	nop	—	—	—	—	11011111	0.013728000
4	0004	0000	irqsvc	nop	—	—	—	—	11011111	0.013860000
5	0004	0000	irqsvc	nop	—	—	—	—	11011111	0.013992000
6	0004	0000	irqsvc	nop	—	—	—	—	11011111	0.014124000
7	0004	0000	irqsvc	nop	—	—	—	—	11011111	0.014256000

Figure A.2: Trace Display - Filter Trace for ISR Time

计算中断开始之间的差数.在图A.2中,差数为132微妙.

方案B: 事件之间的定时

另一个更直接的方法是使用事件之间的定时触发.对于这种触发类型,计时标志发生器保持为零,直到指定的开始事件发生.然后它继续递增,直到指定的停止事件发生.然后它可以用来测量逝去的时间.

要清除所有中断,跟踪和触发点,选择*Debug>Clear All Points*.选择*Debug>Complex Trigger Settings*打开复杂触发设置对话框.

为程序存储器的事件之间计时来设置复杂触发设置对话框.点击**开始定时器**标签.

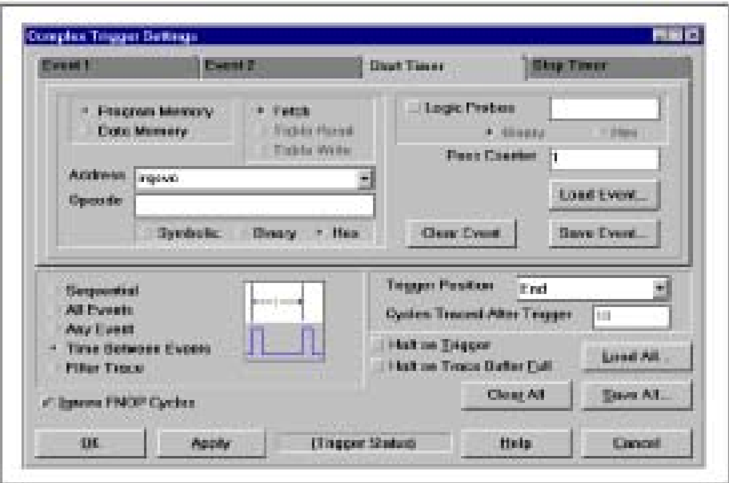


Figure A.3: Start Timer for ISR Time

忽略FNOP周期应被选中.输入ISR开始地址(数字地址或标志符).在图A.3,地址是irqsvc.点击**停止定时器**标签.

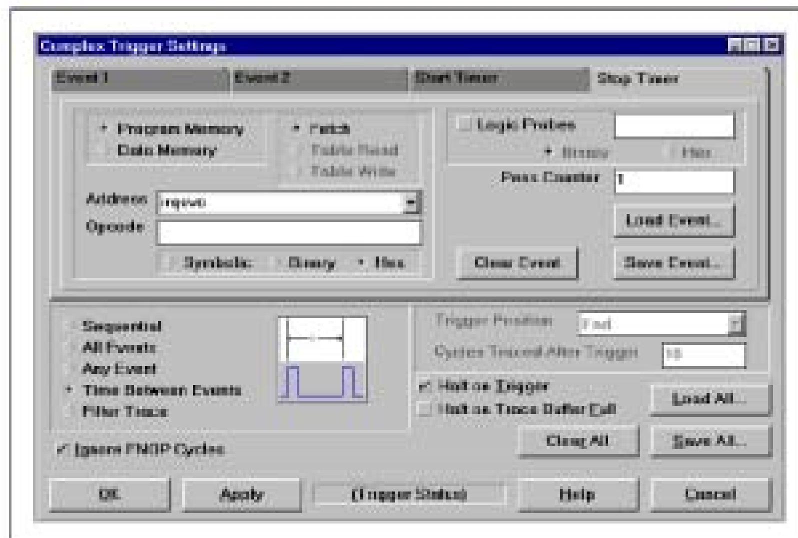


Figure A.4: Stop Timer for ISR Time

再次输入ISR开始地址(数字地址或标志符).在图A.4中,地址仍是irqsvc.选中触发中止.

点击**Apply**或**OK**.

复位系统(*Debug>System Reset*),然后运行程序(*Debug>Run>Run*,或工具栏上绿色灯图标).中止之后,打开跟踪存储器窗口(*Window>Trace Memory*).

Trace Memory Window

File Data Options Help

	Addr	Op	Label	Instruction	SA	SD	DA	DD	Ex	Time
-7	006B	2868		goto delay2	024	7C	024	79	11111111	0.000125000
-6	006C	00A4		(Forced NOP	—	—	—	—	11111111	0.000126000
-5	0068	0000	delay2	nop	—	—	—	—	11111111	0.000127000
-4	0069	0000		nop	—	—	—	—	11111111	0.000128000
-3	006A	0BA4		(Forced NOP	—	—	—	—	11111111	0.000129000
-2	006A	0BA4		(Forced NOP	—	—	—	—	11111111	0.000130000
-1	0004	0000	irqsvc	nop	—	—	—	—	11111111	0.000131000
0	0005	0000		nop	—	—	—	—	111	0.000132000

Figure A.5: Trace Display - TBE for ISR Time

当触发中止执行读时间.这就是中断开始事件之间的时间.和方案A一样,我们看到的时间为132微秒.

A.4.2 变量不正确

一个变量在一段代码不正确.代码显示这个变量被正确的设置,但不知在什么地方这个变量被改变了.那么错误代码在什么地方?

为了指出一个变量的值为什么不正确,你可以使用跟踪显示来捕捉这个值被读或写.依据这点,你可以设置复杂触发来帮助你调试这个问题.

清除所有中断,跟踪和触发点,选中*Debug>Clear All Points*.

图A.6中的程序显示输出变量不对.应该递增num_out的16位值,并将它送给端口.



Figure A.6: Source Code File Window

配置跟踪显示(*Window>Trace Memory,Options>Configure*)为关注源和目标地址和数据.



Figure A.7: Configure Trace for Variable

现在设置一个过滤跟踪来查看进行计算的程序.选中*Debug>Complex Trigger Settings*打开复杂触发设置对话框.



Figure A.8: Filter Trace for Variable

设置复杂触发设置对话框用于程序存储器过滤跟踪.忽略FNOP周期不应被选中.

输入程序地址范围(数字地址或标志符).在图A.1中,地址范围是numout:numout_x.

如果无限事件被选中,取消选择.然后输入捕捉事件的数目,或你想捕捉程序到跟踪存储器窗口的次数.在图A.1中,捕捉事件为200.

点击应用或**OK**.

复位系统(*Debug>System Reset*),然后运行你的程序(*Debug>Run>Run*,工具栏上的绿色灯或**<F9>**).在它中止后,打开跟踪存储器窗口(*Window>Trace Memory*).

Trace Memory Window

	Addr	Label	Instruction	SA	SD	DA	DD
55	005B		goto nxt	num_out	01	num_out	02
56	005C		incf 0x26	—	—	—	—
57	005D	nxt	call numout	—	—	—	—
58	005E	numout_x	return	—	—	—	—
59	005E	numout_x	return	—	—	—	—
60	0058	numout	nop	—	—	—	—
61	0059		nop	—	—	—	—
62	005A		incfsz 0x25	—	—	—	—
63	005B		goto nxt	num_out	02	num_out	03
64	005C		incf 0x26	—	—	—	—
65	005D	nxt	call numout	—	—	—	—
66	005E	numout_x	return	—	—	—	—
67	005E	numout_x	return	—	—	—	—
68	0058	numout	nop	—	—	—	—
69	0059		nop	—	—	—	—
70	005A		incfsz 0x25	—	—	—	—
71	005B		goto nxt	num_out	00	num_out	01

Figure A.9: Trace Display - Filter Trace for Variable

在图A.9中,num_out的值递增为0,1,2,0…有些事情必须设置为0.由于这个值一旦初始化为零,在第二次发生的触发应该捕捉到问题区域.

清除所有中断,跟踪和触发点,选择*Debug>Clear All Points*.

重新打开复杂触发设置对话框(*Debug>Complex Trigger Settings*),在程序存储器设两个顺序事件给相同的地址.

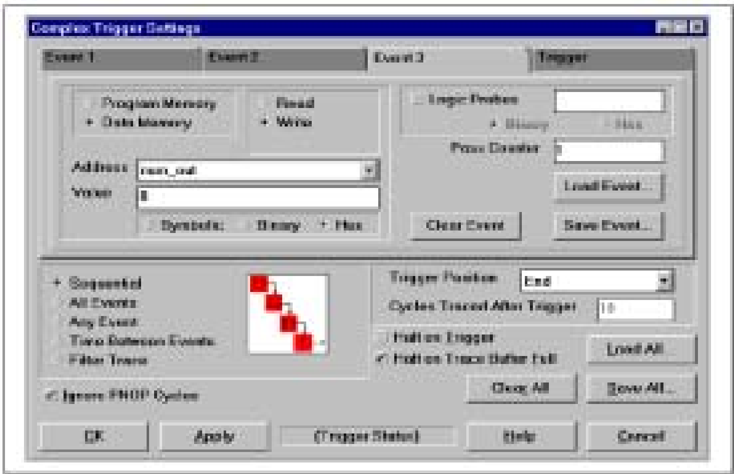


Figure A.10: Sequential Trigger for Variable

点击**事件3**标签.在复杂触发设置对话框上设置为程序存储器顺序事件.选择写.

在地址框输入变量名,数值框输入’0’.在图A.1,变量名是num_out.

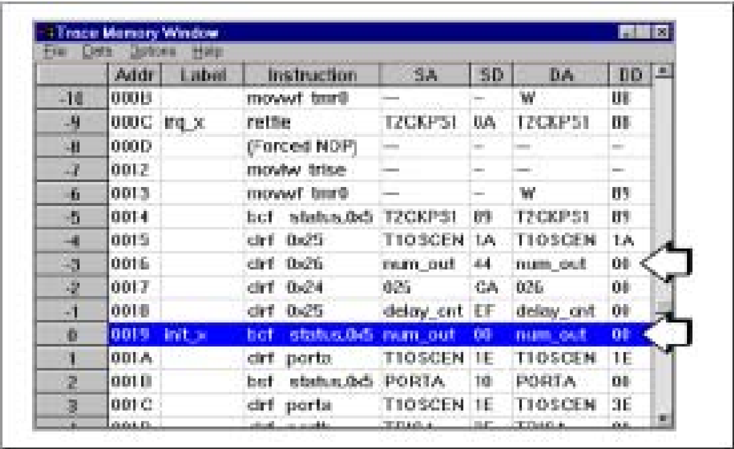
触发位置应该为End.选择跟踪缓冲器满中止和忽略FNOP周期.

点击**触发**标签.再次设置复杂触发设置对话框用于数据存储器顺序事件.选择写.

在地址框输入变量名,数值框输入’0’.

点击**Apply**或**OK**.

复位系统(*Debug>System Reset*),然后运行你的程序(*Debug>Run>Run*,工具栏上的绿色灯或**<F9>**).中止之后,打开跟踪存储器窗口(*Window>Trace Memory*).



	Addr	Label	Instruction	SA	SD	DA	DD
-10	000B		movwf tmr0	—	—	W	08
-9	000C	irq_x	retfie	T2CKPS1	0A	T2CKPS1	08
-8	000D		(Forced NOP)	—	—	—	—
-7	0012		movlw trise	—	—	—	—
-6	0013		movwf tmr0	—	—	W	09
-5	0014		bcf status,0x5	T2CKPS1	09	T2CKPS1	09
-4	0015		clrf 0x25	T1OSCEN	1A	T1OSCEN	1A
-3	0016		clrf 0x26	num_out	44	num_out	00
-2	0017		clrf 0x24	025	CA	025	00
-1	0018		clrf 0x25	delay_cnt	EF	delay_cnt	00
0	0019	init_x	bcf status,0x5	num_out	00	num_out	00
1	001A		clrf porta	T1OSCEN	1E	T1OSCEN	1E
2	001B		bcf status,0x5	PORTA	10	PORTA	00
3	001C		clrf porta	T1OSCEN	1E	T1OSCEN	3E

Figure A.11: Trace Display - Sequential Trigger 1 for Variable

滚动到触发点,(即周期0).这儿num_out设为0.然而你会看到num_out在周期-3也设为0.

由于这只是初始化,需要更多的信息来发现问题.再次运行,当程序中止时,再次观察跟踪窗口的周期0.

Trace Memory Window

	Addr	Label	Instruction	SA	SD	DA	DD
-10	0069		nop	—	—	—	—
-9	006A		decfsz 0x24	—	—	—	—
-8	006B		goto delay2	delay_cnt	02	delay_cnt	01
-7	006C		[Forced NOP]	—	—	—	—
-6	0068	delay2	nop	—	—	—	—
-5	0069		nop	—	—	—	—
-4	006A		decfsz 0x24	—	—	—	—
-3	006B		[Forced NOP]	delay_cnt	01	delay_cnt	00
-2	006C		movwf 0x24	—	—	—	—
-1	006D		decfsz 0x25	delay_cnt	00	delay_cnt	0F
0	006E		[Forced NOP]	num_out	01	num_out	00
1	006F	delay_x	return	—	—	—	—
2	0070		[Forced NOP]	—	—	—	—
3	0041		bsf porte,0x1	—	—	—	—

Figure A.12: Trace Display - Sequential Trigger 2 for Variable

变量num_out在delay2程序中被改写,作为decfsz delay_cnt指令的一个结果.更进一步观察delay_cnt变量,你会发现它是用作一个16位变量,但是在源程序中,它只是设为8位.因此,它与num_out的值交叉了,这样在延时程序中遭到破坏.

改变源代码,以便delay_cnt被设为16位变量(即,改变delay_cnt为delay_cnt:2).

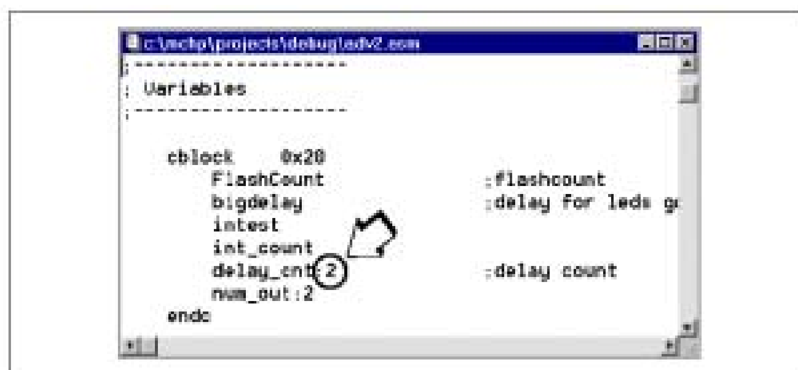


Figure A.13: Source Code File Update

A.4.3 负变量

当一个特定的变量变为一个带符号负值(8位)时,程序就非法执行了.

通过使用一个顺序触发和跟踪显示,你可以看到当变量变为负值时发生了什么.

清除所有中断,跟踪和触发点,选择*Debug>Clear All Points*.配置跟踪显示(*Window>Trace Memory*, *Options>Configure*)到关注源代码和目标地址和数据.



Figure A.14: Configure Trace for Negative Variable

选择*Debug>Complex Trigger Settings*打开复杂触发设置对话框.

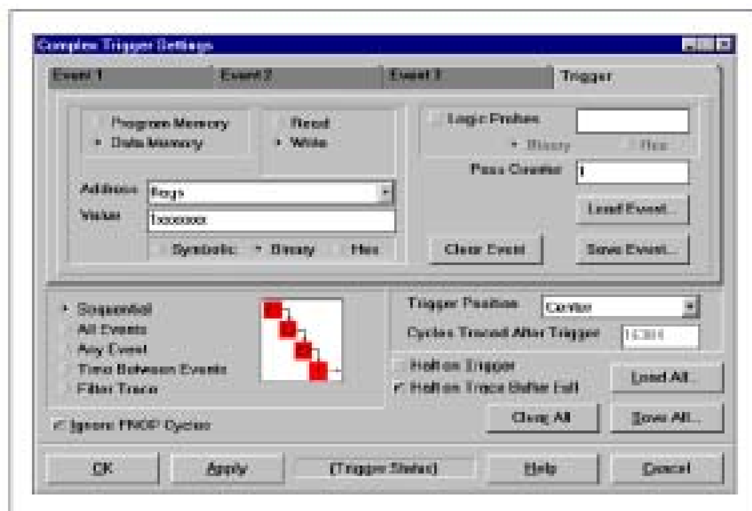


Figure A.15: Sequential Trigger for Negative Variable

设置复杂触发设置对话框用于数据存储器的顺序事件.选择写.

使用8位带符号算术,0-7F为正,80-FF为负.这意味着对于所有负数,位7都为1.因此你可以在这位变为一时触发.

在地址框输入变量名.在图A.1,变量名是flags.然后在数值框输入’1xxxxxxx’,其中’x’意思是不关心的.点击二进制.

触发位置应在中央.选择跟踪缓冲器满中止和忽略FNOP周期.

点击**Apply**或**OK**.

复位系统(*Debug>System Reset*),然后运行你的程序(*Debug>Run>Run*,工具栏上的绿色灯或**<F9>**).中止之后,打开跟踪存储器窗口(*Window>Trace Memory*).

	Addr	Label	Instruction	SA	SD	DA	DD
-10	006D		decfsz 0x25	—	—	—	—
-9	006E		[Forced NOP]	delay_cnt	01	delay_cnt	00
-8	006F		movwf 0x25	—	—	—	—
-7	0070		decfsz 0x26	delay_cnt	00	delay_cnt	00
-6	0071		[Forced NOP]	026	01	026	00
-5	0072	delay_x	return	—	—	—	—
-4	0073		[Forced NOP]	—	—	—	—
-3	0046		bsf porte,0x2	—	—	—	—
-2	0047		decf 0x20	PORTE	03	PORTE	07
-1	0048		incf 0x24	FlashCount	03	FlashCount	02
0	0049	loop_x	goto loopAround	flags	7F	flags	80
1	004A		[Forced NOP]	—	—	—	—
2	002F	loopAround	clrf porta	—	—	—	—
3	0030		clrf portb	RCIF	00	RCIF	00
4	0031		movlw 0xFF	PEIE	01	PEIE	00
5	0032		movwf portc	—	—	W	FF
6	0033		movwf portd	GIE	FF	GIE	FF
7	0034		movwf porte	PORTD	FD	PORTD	FF
8	0035	pcloop	call numout	PORTE	07	PORTE	FF

Figure A.16: Trace Display - Sequential Trigger for Negative Variable

标志值从7F变为80.检查这个触发之前(负值)和之后(正值)的周期来决定你的程序出现了什么问题.

A.4.4 所以代码被执行

测试一个应用,确信所以的代码都被执行.

查看代码覆盖,6.4节.

A.4.5 输出不正确

一个应用输出不正确.你如何发现那儿不正确.

使用逻辑探针在输出值上触发,然后检查在追踪缓冲器里的被执行的代码找到问题。 有两种可能的方法来实现： 使用8 次外部输入(EXT7： EXT0) ,或者使用触发输入(TRGIN)。更多信息请看B.6节.

方案A: 外部输入

清除所有中断,跟踪和触发点,选择*Debug>Clear All Points*. 选择Debug>Complex Trigger Settings打开复杂触发设置对话框.

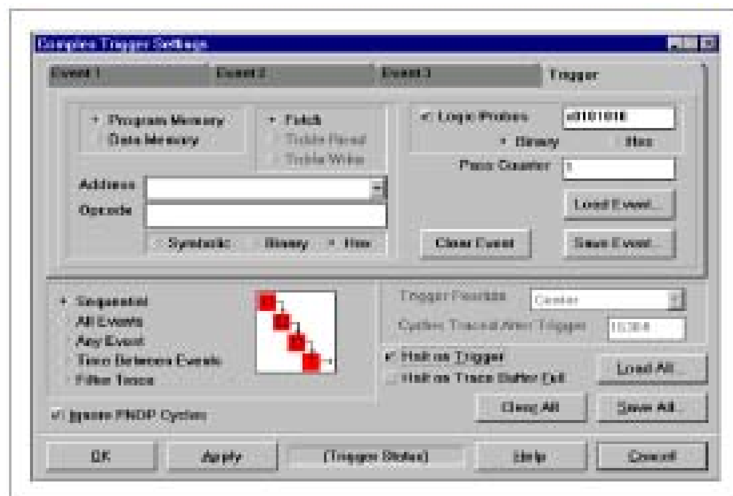


Figure A.17: Sequential Trigger for Logic Probes

设置复杂触发设置对话框为程序存储器顺序跟踪.忽略FNOP周期不应被选中.

选中逻辑探针,并为触发器输入一个8位值.当这个值出现在探针**EXT7:EXT0**上时,触发器会被触发.你可以输入二进制(ex: 00001111)或十六进制(ex: 0x0F).'x'字符表示不用关心.

选中触发中止,然后点击**Apply**或**OK**.

将逻辑探针连接到相应的端口引脚.例如,如果你在PORTB输出触发,那么将探针EXT0连到RB0,EXT1连到RB1,等等.

复位系统(*Debug>System Reset*),然后运行你的程序(*Debug>Run*或工具栏上绿色灯图标).在它中止以后,打开跟踪存储器窗口(*Window>Trace Memory*).

查看被执行的代码来调试你的程序.

方案B: 触发输入

清除所有中断,跟踪和触发点,选择*Debug>Clear All Points*. 选择*Debug>Trigger In/Out Settings*打开复杂触发设置对话框.



Figure A.18: Trigger In for Logic Probes

在外部触发输入选中使能中断,然后选择上升沿或下降沿.如果你想跟踪缓存在触发输入的上升沿凝固,也选择这个选项.点击**OK**.

将触发输入(TRIGIN)探针连接到相应的输出.例如,如果一个错误的输出对应于PORTB的RB2变为高电平,你可以将TRIGIN连到RB2.

复位系统(*Debug>System Reset*),然后运行你的程序.在它中止后,打开跟踪存储器窗口.

查看被执行的代码来调试你的程序.

A.4.6　执行的循环次数

一个应用循环必须被执行一个次数.你如何知道循环是否执行了那么多的次数?

你可以使用通过计数器来决定一个循环是否被执行了一个特定的次数.

清除所有中断,跟踪和触发点,选择*Debug>Clear All Points*.选择*Debug>Complex Trigger Settings*打开复杂触发设置对话框.

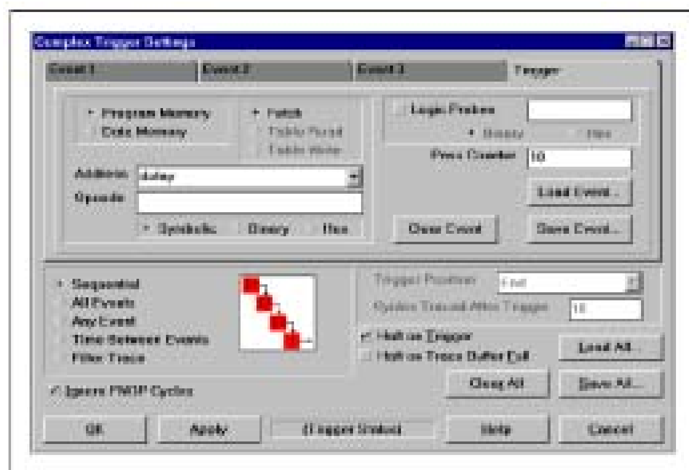


Figure A.19: Sequential Trigger for Pass Count

设置复杂触发设置对话框为程序存储器顺序事件跟踪.忽略FNOP周期应被选中.

输入循环开始的地址(一个数字地址或一个标志符).在图A.1中,地址是delay.

为通过计数器输入一个值.在图A.19中,这个数是10.选中触发中止,然后点击**Apply**或**OK**.

复位系统(*Debug>System Reset*),然后运行你的程序.

如果你的程序中止,那么循环执行的次数至少应为通过计数器显示的数字.如果程序不中止,那么循环就没有执行那么多次数.

A.4.7　虚中断

当一个中断发生时,但是没有中断标志被置位.

如果当进入一个中断服务程序时你中断程序执行,但是却没有标志被置位.这时你就是遇到了虚中断.

当你的程序正在清中断标志时产生了一个中断,虚中断就可能产生.由于PIC单片机流水线结构,当一条清中断标志的指令被取指时,中断可能产生.下一条取指会跳到ISR,但是接着执行的指令会是清中断标志.

附录 B　Pod 电气特性

B.1　介绍

这部分描述了 MPLAB ICE 在线仿真器主机的硬件电气特性.

对于某一特定的处理器模块/设备适配器,参考 MPLAB ICE 处理器模块和设备适配器特性.

对于某一特定转换座的信息,参考 MPLAB ICE 转换座特性.

B.2　要点

这章附录讲述了仿真器主机的下列电气特性:

- 电源
- 并行口
- 指示灯
- 逻辑探头

B.3　电源

MPLAB ICE 系统的电源是通过一个外部+5V 电源供应的.这个输入口位于主机的后侧,如图 B.1 所示.电源开/关开关也位于主机的后侧.

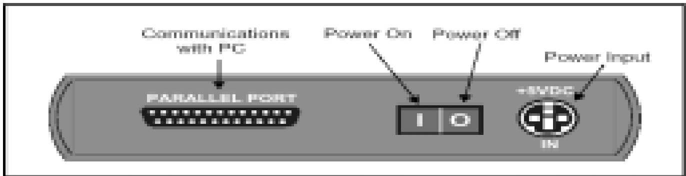


Figure B.1 MPLAB ICE Rear Panel

电源供应要求:

MPLAB ICE 2000: +5V, ±5%,3.0A

WARNING	不要将 PICMASTER-CE 的电源用于 MPLAB ICE.否则会造成损坏.

B.4 并行口

MPLAB ICE 是通过一个工业标准双向并行外围接口连接到主控电脑的.并行口连接器位于主机的后侧面板上.主控电脑要求 **IEEE 1284-A** 类型的连接器.

仿真主机可以连接到任何 **IEEE** 标准 **1284** 并行外围接口,并且符合 **IEEE 1284** 特性标准的机械和电气设备.MPLAB ICE 使用它自身的通讯协议和设备驱动来与电脑进行通讯.

WARNING	使用 MPLAB ICE 时与软件狗,外部磁盘驱动器或其它并行接口设备(即打印机,扫描仪)连接,**会导致这些设备永久性的损坏**.

MPLAB ICE 支持可兼容和双向并口两种模式.MPLAB 会决定电脑配置为哪种模式.更多相关信息请参考 4.4 节.

并行接口信号安装表 B.1 所示使用.关于 A 类型标准并口 PC 接口的完全引脚输出,请参考 **IEEE 1284** 特性.

Table B.1: Parallel Printer Port Signal Assignment Bi-directional Mode

Pin Number	MPLAB ICE Signal	Direction[1]	Description
1	*STROBE	Out	Read/Write Strobe
2-9	Data[0:7]	In/Out	Data Bus
17	*R/W	Out	Selects Read or Write Operation
14	*A/D	Out	Selects Address or Data Operation

[1]From computer

Table B.2: Parallel Printer Port Signal Assignment Compatible Mode

Pin Number	MPLAB ICE Signal	Direction[1]	Description
1	*STROBE	Out	Read/Write Strobe
2-9	Data[0:7]	Out	Data Output Bus
17	*R/W	Out	Selects Read or Write Operation
14	*A/D	Out	Selects Address or Data Operation
15	Data D0/D4	In	Data Input Bus
13	Data D1/D5	In	Data Input Bus
12	Data D2/D6	In	Data Input Bus
11	Data D3/D7	In	Data Input Bus

[1]From computer

B.4.1 电缆长度

电脑到 MPLAB ICE 正确运行的电缆长度经测试为 6 英尺.这个长度的电缆和 MPLAB ICE 一起售出.

B.5 指示灯

关于指示灯的相关信息见表 B.3

Table B.3: Indicator LEDs

LEDs		
Label	Color	Description
P	Green	Power Indicator
E	Red	Processor Module Insertion Error
R	Green	Emulator Run Mode
H	Red	Emulator Halt Mode



Figure B.2 MPLAB ICE Front Panel

B.5.1 电源 LED

LED	状况
亮	系统供电正常
不亮	没有供电或供电不足
闪烁	检测到错误

LED 位于主机前端面板上,当系统供电正常时会被点亮.

如果内部系统电压低于4.65(±5%)V,LED被关闭,表示供电不足.如果发生这种情况,应撤掉电源,直到找出电压不足的原因.

在主机上带一个中断电子电路,如果出现系统错误 LED 会闪烁.闪烁的频率绝对错误的类型.

闪烁频率	意味着
每秒 2 次	过电流(超过 3A)
每秒 8 次	电压过低(小于等于 4.65V)
闪 3 次,停止,闪 3 次	处理器模块电压过低

要将系统从一次错误中复位,可以先关闭主机,然后再打开.如果仍出现问题,请联系 Microchip 支持.

B.5.2 处理器模块插入错误 LED(E)

LED	状况
亮	处理器模块没有完全出入或插入不正确
不亮	处理器模块插入正确

当处理器模块插入不完全或不正确,LED 会被点亮.如果 LED 被点亮,关闭电源,然后拔出并重新插入处理器模块.

B.5.3 仿真器运行模式 LED(R)

LED	状况
亮	处理器正在运行
不亮	处理器被中止
闪烁	处理器执行单步

当处理器在运行时,LED 灯会变亮,当执行单步时,会闪烁.

B.5.4 仿真器中止模式 LED(H)

LED	状况
亮	处理器中止
不亮	处理器在运行

当处理器被中止时,这个 LED 会被点亮.当 MPLAB ICE 被选择作为 MPLAB IDE 开发模式时,这个 LED 和电源 LED 应该被点亮.

这个 LED 的功能是和仿真器运行模式 LED 共同作用的.

B.5.5 鬼灯

有可能系统没有上电,但你可能会看到运行模式 LED 被点亮.这是由于并行口上的有源信号会有缓冲偏移,并有足够的电压点亮 LED.这并不是什么问题,也不会对任何部件造成损坏.

B.6 逻辑探头

MPLAB ICE 仿真器前端面板的 14 脚连接器(图 B.3)提供了电源,地,和外部中断输入,一个触发输入,一个触发输出和最多 8 个跟踪/触发输入.

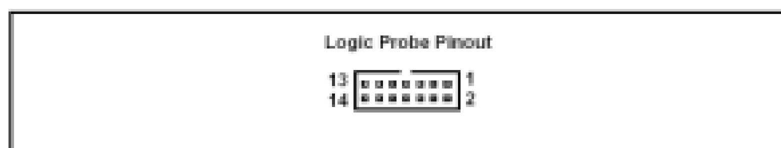


Figure B.3 Logic Probe Pinout

逻辑探针是与连接器相匹配的.具体功能见表 B.4.探针通过颜色来区分功能.

Table B.4: Logic Probe Pinout Description

Pin	I/O	Name	Function	Color
1	O	V_{DD}	System Power. Will supply +5V ±5%, up to 250 mA	Red
2	O	HLTOUT	Processor Halted Signal. Indicates whether the processor is halted (high) or running (low).	Gray
3	O	TRGOUT	Trigger/Break Out	Gray
4	I	TRGIN	Trigger In. Active-high input will freeze the trace buffer without halting the processor or edge triggered input will halt the processor.	Gray
5	I	EXT7	External input bit 7 of the trace/trigger inputs.	White
6	I	EXT6	External input bit 6 of the trace/trigger inputs.	White
7	I	EXT5	External input bit 5 of the trace/trigger inputs.	White
8	I	EXT4	External input bit 4 of the trace/trigger inputs.	White
9	I	EXT3	External input bit 3 of the trace/trigger inputs.	White
10	I	EXT2	External input bit 2 of the trace/trigger inputs.	White
11	I	EXT1	External input bit 1 of the trace/trigger inputs.	White
12	I	EXT0	External input bit 0 of the trace/trigger inputs.	White
13	Gnd	GND	System Ground	Black
14	Gnd	GND	System Ground	Black

逻辑探头的电气特性见表 B.5

Table B.5: Logic Probe Electrical Specifications

Logic Inputs	V_{IH} = 3.2V min
	V_{IL} = 1.8V max
Logic Outputs	V_{OH} = 2.4V min
	V_{OL} = 0.4V max
TRGIN	Minimum input pulse width = 15 nsec
TRGOUT	The output pulse width of the filter trace is one bus cycle, which is ¼ the clock speed (PICmicro MCU devices use four clocks for each instruction).

附录 C　从 PICMASTER 进行移植

C.1　介绍

尽管 MPLAB ICE 和 PICMASTER 通过 MPLAB IDE 有相同的基本外观,但也有些差异.这部分附录描述了那些差异.

C.2　要点

这部分附录包括:

- 不用更改的项目
- 硬件设置
- 如何记时事件
- 设置断点
- 复杂触发与中断/跟踪/触发点
- 外部输入输出

C.3　不用更改的项目

许多MPLAB ICE功能操作和PICMASTER是一样的.这些功能包括:

- 项目，包括语言工具和接口,以及MPLAB IDE编辑器
- 查看程序存储器,校验值和数据(EEPROM)存储器(如果有的话),文件寄存器和SFR's.
- 修改程序存储器,校验值和数据(EEPROM)存储器(如果有的话),文件寄存器和SFR's.
- 命名断电
- 运行,中止,单步,单步跳跃,复位.

C.4　硬件设置

PICMASTER探针包含各种DIP开关和跳线来配置探针.这些选项现在是可以通过软件来选择的.可以访问 *Options>Development Mode*,选择**Power**和**Clock**标签.

PICMASTER跳线和开关可以通过开发模式对话框下列选项来选择的.

Hardware Configuration Items		
Description	**PICMASTER**	**MPLAB ICE**
Emulation Power	Jumper (+5VSYS/+5VEXT)	Options>Development Mode, **Power** tab
Clock Source	Jumper (EXTCLK/INTCLK)	Options>Development Mode, **Clock** tab
Clock Frequency	Oscillator installed on probe	Options>Development Mode, **Clock** tab
Oscillator Type	DIP switches	Options>Development Mode, **Clock** tab

C.5 如何进行事件记时

使用PICMASTER,事件可以通过Stopwatch对话框来记时.而MPLAB ICE,事件是通过使用跟踪存储器窗口的时间标志来记时的.

至少有两种方法来对一个事件进行记时:

1. 在事件的开始处设置程序计数器来记时.在事件的结束处设置一个软件断点.然后运行,等待执行中止,查看跟踪缓存器.最后的时间标志就是这个事件之间的时间.
2. 使用一个事件之间的触发记时,设置开始定时器和停止定时器事件,然后运行. 最后的时间标志就是这个事件之间的时间.使用这种方式,不用中止处理器.

C.6 设置断点

下面的表格显示了断点类型和三种仿真器的能力.

Break Point Comparison		
Type	**PICMASTER**	**MPLAB ICE**
Software	None	Unlimited
Hardware	Unlimited	Unlimited

软件断点和硬件断点不同的是,软件断点没有执行偏移.如果一个断点设在程序存储器地址100,然后处理器在地址100的指令被执行之前就被中止了.当仿真中止时,程序计数器值会是100.使用硬件断点,至少位于100的这条指令会被执行.根据硬件断点所设置的条件,一条或更多条指令会被执行.

MPLAB ICE仿真器的硬件断点通过复杂触发对话框来设置.单个硬件断点也可以通过鼠标右键来设置.

C.6.1 复杂触发与中断/跟踪/触发点

PICMASTER允许你指定每个程序存储器地址作为零或下列选项:

- 一个断点,执行它时终止仿真

• 一个跟踪点,在跟踪窗口会出现它的执行
• 一个触发点,执行它时,会产生一个外部信号

MPLAB ICE保持了这个功能,但是是通过一个不同的方式.软件断点是通过断点设置对话框或鼠标右键菜单来设置的.硬件断点是通过在复杂触发对话框指定触发中止,或鼠标右键来设置的.这两种方式都会中止仿真.

在MPLAB ICE中，跟踪稍微有些不同.默认情况下，所以程序执行都是被跟踪的．只需打开跟踪存储器窗口查看最后执行的跟踪情况.如果希望只跟踪某一特定的事件,可以使用一个过滤跟踪触发来过滤跟踪.

在MPLAB ICE中,触发点可以通过使用一个过滤跟踪复杂触发在期望的操作上产生一个触发来获得.然后触发输入/输出设置对话框(*Debug>Trigger In/Out Settings*)可以用于产生一个外部信号脉冲,只要复杂触发事件的其中一个发生.

C.7 外部输入输出

PICMASTER和MPLAB ICE逻辑探针的相似和不同的地方列出在下表中:

PICMASTER vs. MPLAB ICE I/O			
PICMASTER	**MPLAB ICE**	**I/O**	**Description**
VDD	VDD	O	+5V power (250mA Max)
GND	GND	I	Common ground
Tr[0:7]	EXT[0:7]	I	Eight (8) external trace inputs
BRKT	TRGIN	I	External Trace Halt Signal. Halts the trace buffer on a rising edge without halting the processor.
BRK	TRGIN	I	External Break Input Signal. Halts the processor either on a rising or falling edge, where the edge is software programmable.
TRIGO	TRGOUT*	O	Trigger Output Signal. Use this signal for triggering oscilloscopes, accurate time measurement, etc.
TRIGT	—	O	Trace Trigger Signal. This is the same clock applied to the trace buffer for incrementing trace address. Use this clock to trigger instruments like logic analyzers.
—	HLTOUT	O	Processor Halted Signal. Indicates whether the processor is halted (high) or running (low).
* To get continuous triggers from MPLAB ICE 2000 like the PICMASTER triggers, you should set up a filtered trace with Infinite Events and check the box on Trigger In/Out Dialog.			